المملكة العربية السعودية

وزارة التعليم
Ministry of Education

قررت وزارة التعليم تدريس
هذا الكتاب وطبعه على نفقتها

# الذكاء الاصطناعي

## التعليم الثانوي - نظام المسارات

## السنة الثالثة

طبعة 2025-1447

يوزع مجاناً ولا يباع
وزارة التعليم
Ministry of Education
2025 - 1447

ح المركز الوطني للمناهج، ١٤٤٦ هـ

المركز الوطني للمناهج
الذكاء الاصطناعي - المرحلة الثانوية - نظام المسارات - السنة الثالثة.
المركز الوطني للمناهج. - الرياض، ١٤٤٦ هـ
٣٣٩ ص ؛ ٢١ x ٢٥,٥ سم

رقم الإيداع : ١٨٧٣٠ / ١٤٤٦
ردمك :١-١٧-٥١٤-٦٠٣-٩٧٨

حقوق الطبع والنشر محفوظة لوزارة التعليم
www.moe.gov.sa

مواد إثرائية وداعمة على "منصة عين الإثرائية"

ien.edu.sa

أعزاءنا المعلمين والمعلمات، والطلاب والطالبات، وأولياء الأمور، وكل مهتم بالتربية والتعليم:
يسعدنا تواصلكم؛ لتطوير الكتاب المدرسي، ومقترحاتكم محل اهتمامنا.

fb.ien.edu.sa

وزارة التعليم
Ministry of Education
2025 - 1447

الناشر: شركة تطوير للخدمات التعليمية

تم النشر بموجب اتفاقية خاصة بين شركة Binary Logic SA وشركة تطوير للخدمات التعليمية
(عقد رقم 2022/0003) للاستخدام في المملكة العربية السعودية

حقوق النشر © Binary Logic SA 2025

جميع الحقوق محفوظة. لا يجوز نسخ أي جزء من هذا المنشور أو تخزينه في أنظمة استرجاع البيانات أو نقله بأي شكل أو بأي وسيلة إلكترونية أو ميكانيكية أو بالنسخ الضوئي أو التسجيل أو غير ذلك دون إذن كتابي من الناشرين.

يُرجى ملاحظة ما يلي: يحتوي هذا الكتاب على روابط إلى مواقع إلكترونية لا تُدار من قبل شركة Binary Logic. ورغم أنَّ شركة Binary Logic تبذل قصارى جهدها لضمان دقة هذه الروابط وحداثتها وملاءمتها، إلا أنها لا تتحمل المسؤولية عن محتوى أي مواقع إلكترونية خارجية.

إشعار بالعلامات التجارية: أسماء المنتجات أو الشركات المذكورة هنا قد تكون علامات تجارية أو علامات تجارية مُسجَّلة وتُستخدم فقط بغرض التعريف والتوضيح وليس هناك أي نية لانتهاك الحقوق. تنفي شركة Binary Logic وجود أي ارتباط أو رعاية أو تأييد من جانب مالكي العلامات التجارية المعنيين. تُعد Tinkercad علامة تجارية مُسجَّلة لشركة Autodesk Inc. تُعد Python وشعارات Python علامات تجارية مسجلة لشركة Python Software Foundation. تُعد Jupyter علامة تجارية مُسجَّلة لشركة Project Jupyter. تُعد CupCarbon علامة تجارية مُسجَّلة لشركة CupCarbon. تُعد Arduino علامة تجارية مُسجَّلة لشركة Arduino SA. تُعد Webots علامة تجارية مُسجَّلة لشركة Cyberbotics Ltd.

ولا ترعى الشركات أو المنظمات المذكورة أعلاه هذا الكتاب أو تصرح به أو تصادق عليه.

حاول الناشر جاهدًا تتبع ملاك الحقوق الفكرية كافة، وإذا كان قد سقط اسم أيٍّ منهم سهوًا فسيكون من دواعي سرور الناشر اتخاذ التدابير اللازمة في أقرب فرصة.

binarylogic

# مقدمة

إن تقدم الدول وتطورها يقاس بمدى قدرتها على الاستثمار في التعليم، ومدى استجابة نظامها التعليمي لمتطلبات العصر ومتغيراته. وحرصًا من وزارة التعليم على ديمومة تطوير أنظمتها التعليمية، واستجابة لرؤية المملكة العربية السعودية 2030 فقد بادرت الوزارة إلى اعتماد نظام «مسارات التعليم الثانوي» بهدف إحداث تغيير فاعل وشامل في المرحلة الثانوية.

إن نظام مسارات التعليم الثانوي يقدم أنموذجًا تعليميًا متميزًا وحديثًا للتعليم الثانوي بالمملكة العربية السعودية يسهم بكفاءة في:

- تعزيز قيم الانتماء لوطننا المملكة العربية السعودية، والولاء لقيادته الرشيدة حفظهم الله، انطلاقًا من عقيدة صافية مستندة على التعاليم الإسلامية السمحة.
- تعزيز قيم المواطنة من خلال التركيز عليها في المواد الدراسية والأنشطة، اتساقًا مع مطالب التنمية المستدامة، والخطط التنموية في المملكة العربية السعودية التي تؤكد على ترسيخ ثنائية القيم والهوية، والقائمة على تعاليم الإسلام والوسطية.
- تأهيل الطلبة بما يتوافق مع التخصصات المستقبلية في الجامعات والكليات أو المهن المطلوبة؛ لضمان اتساق مخرجات التعليم مع متطلبات سوق العمل.
- تمكين الطلبة من متابعة التعليم في المسار المفضل لديهم في مراحل مبكرة، وفق ميولهم وقدراتهم.
- تمكين الطلبة من الالتحاق بالتخصصات العلمية والإدارية النوعية المرتبطة بسوق العمل، ووظائف المستقبل.
- دمج الطلبة في بيئة تعليمية ممتعة ومحفزة داخل المدرسة قائمة على فلسفة بنائية، وممارسات تطبيقية ضمن مناخ تعليمي نشط.
- نقل الطلبة عبر رحلة تعليمية متكاملة بدءًا من المرحلة الابتدائية حتى نهاية المرحلة الثانوية، وتُسهِّل عملية انتقالهم إلى مرحلة ما بعد التعليم العام.
- تزويد الطلبة بالمهارات التقنية والشخصية التي تساعدهم على التعامل مع الحياة، والتجاوب مع متطلبات المرحلة.
- توسيع الفرص أمام الطلبة الخريجين عبر خيارات متنوعة إضافة إلى الجامعات مثل: الحصول على شهادات مهنية، والالتحاق بالكليات التطبيقية، والحصول على دبلومات وظيفية.

ويتكون نظام المسارات من تسعة فصول دراسية تُدرّس في ثلاث سنوات، تتضمن سنة أولى مشتركة يتلقى فيها الطلبة الدروس في مجالات علمية وإنسانية متنوعة، تليها سنتان تخصصيتان، يُسكّن الطلبة بها في مسار عام وأربعة مسارات تخصصية تتسق مع ميولهم وقدراتهم، وهي: المسار الشرعي، مسار إدارة الأعمال، مسار علوم الحاسب والهندسة، مسار الصحة والحياة، وهو ما يجعل هذا النظام هو الأفضل للطلبة من حيث:

- وجود مواد دراسية جديدة تتوافق مع متطلبات الثورة الصناعية الرابعة والخطط التنموية، ورؤية المملكة 2030، تهدف لتنمية مهارات التفكير العليا وحل المشكلات، والمهارات البحثية.
- برامج المجال الاختياري التي تتسق مع احتياجات سوق العمل وميول الطلبة، حيث يُمكّن الطلبة من الالتحاق بمجال اختياري محدد وفق مصفوفة مهارات وظيفية محددة.
- مقياس ميول يضمن تحقيق كفاءة الطلبة وفاعليتهم، ويساعدهم في تحديد اتجاهاتهم وميولهم، وكشف مكامن القوة لديهم، مما يعزز من فرص نجاحهم في المستقبل.
- العمل التطوعي المصمم للطلبة خصيصًا بما يتسق مع فلسفة النشاط في المدارس، ويعد أحد متطلبات التخرج؛ مما يساعد على تعزيز القيم الإنسانية، وبناء المجتمع وتنميته وتماسكه.
- التجسير الذي يمكن الطلبة من الانتقال من مسار إلى آخر وفق آليات محددة.
- حصص الإتقان التي يتم من خلالها تطوير المهارات وتحسين المستوى التحصيلي، من خلال تقديم حصص إتقان إثرائية وعلاجية.

وزارة التعليم
Ministry of Education
2025 - 1447

- خيارات التعليم المدمج، والتعلم عن بعد، والذي بُني في نظام المسارات على أسس من المرونة، والملاءمة والتفاعل والفعالية.
- مشروع التخرج الذي يساعد الطلبة على دمج الخبرات النظرية مع الممارسات التطبيقية.
- شهادات مهنية ومهارية تمنح للطلبة بعد إنجازهم مهامَّ محددة، واختبارات معينة بالشراكة مع جهات تخصصية.

وبالتالي فإن مسار علوم الحاسب والهندسة كأحد المسارات المستحدثة في المرحلة الثانوية يسهم في تحقيق أفضل الممارسات عبر الاستثمار في رأس المال البشري، وتحويل الطالب إلى فرد مشارك ومنتج للعلوم والمعارف، مع إكسابه المهارات والخبرات اللازمة لاستكمال دراسته في تخصصات تتناسب مع ميوله وقدراته أو الالتحاق بسوق العمل.

وتعد مادة الذكاء الاصطناعي أحد المواد الرئيسة في مسار علوم الحاسب والهندسة، حيث تسهم في توضيح مفاهيم الذكاء الاصطناعي والتقنيات المرتبطة بها بما يساعد على توظيف هذه التقنيات في عدة مجالات حياتية مثل المدن الذكية والتعليم والزراعة والطب وغيرها من المجالات الاقتصادية المتنوعة. وتهدف المادة إلى تعريف الطالب بأهمية الذكاء الاصطناعي ودوره في الجيل الرابع من الصناعة. وكذلك تركز على اللبنات الأساسية لتقنيات الذكاء الاصطناعي، ثم تتعرَّض بشكل تفصيلي للتطبيقات المتقدمة التي تتعلق بالأنظمة القائمة على القواعد وأنظمة معالجة اللغات الطبيعية. كما تشتمل هذه المادة على مشاريع وتمارين تطبيقية لما يتعلمه الطالب؛ لحل مشاكل واقعية تحاكي مستوياته المعرفية، بتوجيه وإشراف من المعلم.

ويتميز كتاب الذكاء الاصطناعي بأساليب حديثة، تتوافر فيه عناصر الجذب والتشويق، والتي تجعل الطلبة يقبلون على تعلمه والتفاعل معه، من خلال ما يقدمه من تدريبات وأنشطة متنوعة، كما يؤكد هذا الكتاب على جوانب مهمة في تعليم الذكاء الاصطناعي وتعلمه، تتمثل في:

- الترابط الوثيق بين المحتويات والمواقف والمشكلات الحياتية.
- تنوع طرائق عرض المحتوى بصورة جذابة ومشوقة.
- إبراز دور المتعلم في عمليات التعليم والتعلم.
- الاهتمام بترابط محتوياته مما يجعل منه كلًّا متكاملًا.
- الاهتمام بتوظيف التقنيات المناسبة في المواقف المختلفة.
- الاهتمام بتوظيف أساليب متنوعة في تقويم الطلبة بما يتناسب مع الفروق الفردية بينهم.

ولمواكبة التطورات العالمية في هذا المجال، فإن كتاب مادة الذكاء الاصطناعي سوف يوفر للمعلم مجموعة متكاملة من المواد التعليمية المتنوعة التي تراعي الفروق الفردية بين الطلبة، بالإضافة إلى البرمجيات والمواقع التعليمية، التي توفر للطلبة فرصة توظيف التقنيات الحديثة والتواصل المبني على الممارسة؛ مما يؤكد دوره في عملية التعليم والتعلم.

ونحن إذ نقدم هذا الكتاب لأعزائنا الطلبة، نأمل أن يستحوذ على اهتمامهم، ويُلبي متطلباتهم، ويجعل تعلّمهم لهذه المادة أكثر متعة وفائدة.

والله ولي التوفيق

وزارة التعليم
Ministry of Education
2025 - 1447

بسم الله الرحمن الرحيم

وزارة التعليم
Ministry of Education
2025 - 1447

# الفهرس


## الجزء الأول

### 1. أساسيات الذكاء الاصطناعي ......... 10

الدرس الأول

مقدمة في الذكاء الاصطناعي ........................... 11

تمرينات ............................................. 21

الدرس الثاني

هياكل البيانات في الذكاء الاصطناعي ........... 23

تمرينات ............................................. 50

الدرس الثالث

هياكل البيانات غير الخطيَّة ......................... 53

تمرينات ............................................. 63

المشروع ............................................. 68

### 2. خوارزميات الذكاء الاصطناعي....... 70

الدرس الأول

الاستدعاء الذاتي ................................... 71

تمرينات ............................................. 77

الدرس الثاني

خوارزمية البحث بأولوية العمق
والبحث بأولوية الاتساع ............................ 79

تمرينات ............................................. 86

الدرس الثالث

اتخاذ القرار القائم على القواعد .................. 89

تمرينات ............................................. 105

الدرس الرابع

خوارزميات البحث المستنيرة ..................... 107

تمرينات ............................................. 128

المشروع ............................................. 130

### 3. معالجة اللغات الطبيعية ............. 132

الدرس الأول

التعلُّم الموجَّه ....................................... 133

تمرينات ............................................. 152

الدرس الثاني

التعلُّم غير الموجَّه ................................... 154

تمرينات ............................................. 170

الدرس الثالث

توليد النص ......................................... 172

تمرينات ............................................. 189

المشروع ............................................. 192

## الجزء الثاني

### 4. التعرّف على الصور .................. 196

الدرس الأول

التعلُّم الموجَّه لتحليل الصور ........................ 197

تمرينات ............................................. 218

الدرس الثاني

التعلُّم غير الموجَّه لتحليل الصور .................... 220

تمرينات ............................................. 234

الدرس الثالث

توليد البيانات المرئية ............................... 236

تمرينات ............................................. 246

المشروع ............................................. 248

### 5. خوارزميات التحسين واتخاذ القرار... 250

الدرس الأول

مشكلة تخصيص الموارد .............................. 251

تمرينات ............................................. 264

الدرس الثاني

مشكلة جدولة الموارد ................................ 267

تمرينات ............................................. 279

الدرس الثالث

مشكلة تحسين المسار ................................ 283

تمرينات ............................................. 294

المشروع ............................................. 298

### 6. الذكاء الاصطناعي والمجتمع ....... 300

الدرس الأول

مقدمة في أخلاقيات الذكاء الاصطناعي ............. 301

تمرينات ............................................. 310

الدرس الثاني

التطبيقات الروبوتية 1 ............................... 312

تمرينات ............................................. 326

الدرس الثالث

التطبيقات الروبوتية 2 ............................... 328

تمرينات ............................................. 336

المشروع ............................................. 338



وزارة التعليم
Ministry of Education
2025 - 1447

# الجزء الأول

## الوحدة الأولى
أساسيات الذكاء الاصطناعي

## الوحدة الثانية
خوارزميات الذكاء الاصطناعي

## الوحدة الثالثة
معالجة اللغات الطبيعية

وزارة التعليم
Ministry of Education
2025 - 1447

وزارة التعليم
Ministry of Education
2025 - 1447

# 1. أساسيات الذكاء الاصطناعي

سيتعرّف الطالب في هذه الوحدة على تاريخ الذكاء الاصطناعي (Artificial Intelligence - AI) وتطبيقاته. كما سيتعلّم المزيد حول هياكل البيانات المتقدمة، مثل الطوابير، والمُكدّسات، والقوائم المترابطة، والمُخطّطات، والأشجار الثنائية، وسيَستخدِم هذه التراكيب لاحقًا لإنشاء مشاريع الذكاء الاصطناعي.

## أهداف التعلُّم

بنهاية هذه الوحدة سيكون الطالب قادرًا على أن:

- يَذكُر معالم تاريخ الذكاء الاصطناعي (AI).
- يُعَدِّد أمثلة لتطبيقات الذكاء الاصطناعي (AI).
- يَصِف عمليات هيكل بيانات المُكدّس.
- يَصِف عمليات هيكل بيانات الطابور.
- يُحدِّد الاختلافات بين هيكل بيانات المُكدّس وهيكل بيانات الطابور.
- يَصِف العمليات الرئيسة المُطبَّقة على البيانات في القائمة المترابطة.
- يَشرح استخدام هيكل بيانات الشجرة.
- يُحدِّد الاختلافات بين هيكل بيانات الشجرة وهيكل بيانات المُخطَّط.
- يَستخدِم لغة برمجة البايثون (Python) لاستكشاف هياكل البيانات المُعقَّدة.

## الأدوات

- مفكّرة جوبيتر (Jupyter Notebook)

وزارة التعليم
Ministry of Education
2025 - 1447

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الأول

# مقدمة في الذكاء الاصطناعي

## ما الذكاء الاصطناعي؟

## What is Artificial Intelligence (AI)

**الذكاء الاصطناعي** (AI) هو أحد مجالات علوم الحاسب الآلي التي تُعنى بتصميم وتطبيق البرامج القادرة على محاكاة القدرات المعرفية البشرية. تُظهر هذه البرامج الخصائص التي تَصِف السلوك البشري عادةً، مثل حل المشكلات، والتعلُّم، وصُنع القرارات، والاستدلال، والتخطيط، واتخاذ القرارات، إلخ.

### وكلاء الذكاء الاصطناعي (AI Agents):

وكيل الذكاء الاصطناعي هو برنامج يعمل نيابةً عن المُستخدِم أو النظام في إدراك بيئته، وصنع القرارات، واتخاذ الإجراءات وفقًا لها، وقد يكون الوكيل بسيطًا أو مُعقدًا، ذاتي التحكّم أو شبه ذاتي التحكّم، أو يعمل في بيئات متنوعة، مثل المُستندة إلى الويب، أو الماديّة، أو الافتراضيّة.

### الشبكات العصبية (Neural Networks):

الشبكات العصبية هي نوع من برامج الحاسب المُصمَّمة لمحاكاة طريقة عمل الدماغ البشري، وهي مكونة من خلايا وطبقات عصبية يمكنها معالجة المعلومات ونقلها.

شكل 1.1: بعض مجالات الذكاء الاصطناعي


وزارة التعليم
Ministry of Education
2025 - 1447

## الذكاء الاصطناعي والمجالات الأخرى AI and Other Fields

يرتبط **الذكاء الاصطناعي** (AI) ارتباطًا وثيق الصلة بعدة مجالات أخرى تشمل:

**الفلسفة** (Philosophy): هي أصل العلوم الحديثة، وتُعنى بدراسة المشكلات التي تمثّل أُسس الذكاء الاصطناعي، مثل أصل المعرِفة وتمثيلها، والاستدلال المُستنِد إلى القواعد والمنطق، والتحليل القائم على الأهداف، والصلة بين المعرِفة والتصرُّف.

**الرياضيات** (Mathematics): هي جوهر الذكاء الاصطناعي، حيث تُقدم له لبِنات البناء الأساسية مثل: المنطق، والحوسبة، ونظرية الاحتمالات.

**نظرية القرار** (Decision Theory): تُعنى بدراسة الخصائص المنطقية والرياضية لعملية صُنع القرار، حيث تحلّل عمليّة اتخاذ القرارات في نظام تكون فيه بيئة القرار غير واضحة، وتُطبَّق الأُطُر والأساليب النظرية في هذا المجال باستمرار لحلّ مشكلات الذكاء الاصطناعي.

**علم الأعصاب** (Neuroscience): يُعنى بدراسة الجهاز العصبي البشري، وقد توصل علم الأعصاب إلى نتيجة رئيسة عملت كمبدأ إرشادي للذكاء الاصطناعي، وهي أن مجموعة من الخلايا البسيطة يمكن أن تؤدي إلى نتائج مُعقدة مثل: الفكر، والعمل، والوعي. كما أن الشبكات العصبية الاصطناعية تُحاكي البُنيات العصبية الموجودة في الدماغ البشري.

**علم النفس المعرفي** (Cognitive Psychology): هو أحد فروع علم النفس، ويُعنى بدراسة طريقة تفكير البشر. ولطالما كان الفضل في تحقيق الانجازات والتقدّم في مجال الذكاء الصناعي راجعًا إلى الاكتشافات التي تمّ تحقيقها في هذا المجال، والتي ساعدت على توفير الرؤى التي تساعد أجهزة الحاسب على محاكاة التفكير البشري.

**علوم الحاسب والهندسة** (Computer Science and Engineering): تُعدُّ علوم الحاسب والهندسة حجر الأساس لتوفير البرمجيات والأجهزة اللازمة للذكاء الاصطناعي للانتقال من المبادئ النظرية إلى التطبيقات العملية. وقد واكَب التقدم في الذكاء الاصطناعي باستمرار التطورات في أنظمة التشغيل، والبرمجة، واللغات، والسعة التخزينية، والذاكرة، وقوة مُعالجة البيانات.

**علم التحكّم الآلي** (Cybernetics): يُعنى بدراسة الأنظمة التي تحقق الحالة المرجوة باستلام المعلومات من بيئتها وتعديل سلوكها وفقًا لذلك. الفرق الرئيس بين علم التحكّم الآلي وبين الذكاء الاصطناعيّ هو أن الأول يستخدِم الرياضيات لنمذجة الأنظمة المُغلقة التي يمكن وصفها بالكامل باستخدام متغيرات مُحدَّدة، بينما يستخدِم الذكاء الاصطناعي الاستدلال المنطقي والحوسبة للتغلب على هذه القيود ودراسة المشكلات المُعقَّدة مثل: فهم اللغة والمعلومات المرئية وتوليدهما.

**علم اللُغويات** (Linguistics): هو الدراسة العلمية للغة البشرية، فلطالما كان فهم اللغة البشريّة وتوليدها مجالًا رئيسًا في تطبيقات الذكاء الاصطناعي، كما أدى إلى نشوء حقول فرعية مثل: **معالجة اللغات الطبيعية** (Natural Language Processing - NLP) و**اللغويات الحاسوبية** (Computational Linguistics).

**علم الرؤية** (Vision Science): هو الدراسة العلمية للإدراك البصري. ويُعدّ تعليم أجهزة الحاسب كيفية فهم الصور، والرسوم المتحركة، ومقاطع الفيديو وتوليدها أحد أكثرِ تطبيقات الذكاء الاصطناعي إثارة، وتحديدًا في المجالات الفرعية للتعلُّم العميق ورؤية الحاسب.

**معلومة**

**استُخدِم مصطلح الذكاء الاصطناعي رسميًا للمرة الأولى في عام 1956، مما يجعله أحد أحدث المجالات العلمية نسبيًا.**

## اختبار تورنغ Turing Test

قد يكون اختبار تورنغ هو الطريقة الأكثر شهرة لتعريف الذكاء الاصطناعي، ويعود تاريخ اقتراحه إلى عام 1950، حيث أجرى العالم تورنغ تجربة لمعرفة ما إذا كان الحاسب ذكيًا أم لا.

وأثناء الاختبار، يتوجب على الحاسب أن يجيب عن بعض الأسئلة المكتوبة التي يقدمها **المُوجِّه البشري** (Human Respondent). يُعدُّ الاختبار ناجحًا إذا لم يتمكن المُوجِّه من معرفة ما إذا كانت الإجابة مكتوبة بواسطة إنسان أم بواسطة الحاسب.

لاجتياز الاختبار بنجاح، يجب أن يتمتع الحاسب بالإمكانات الموضحة في الجدول التالي:

**اختبار تورنغ (Turing Test):**
يقيس اختبار تورنغ قدرة الآلة على إظهار سلوك ذكي مكافئ لسلوك الإنسان أو غير قابل للتمييز عنه.

شكل 1.2: تمثيل اختبار تورنغ


وزارة التعليم
Ministry of Education
2025 - 1447

**جدول 1.1: إمكانات الحاسب لاجتياز اختبار تورنغ**

| | |
|---|---|
| 1 | معالجة اللغات الطبيعية؛ لتمكين الحاسب من فهم الأسئلة والرد عليها. |
| 2 | تمثيل المعرفة لتنظيم المعلومات وتخزينها واسترجاعها خلال أداء الاختبار. |
| 3 | الاستدلال المُؤتمت؛ لاستخدام المعلومات المُخزَّنة للإجابة عن الأسئلة. |
| 4 | تعلُّم الآلة للتكيّف مع هياكل اللغات الجديدة مثل: بناء جُمل مختلفة، أو إيجاد مفردات لغوية مختلفة، لم يرها من قبل، أو ليست مخزّنة ضمن المعلومات. |
| 5 | رؤية الحاسب؛ حتى يتمكن من الاستجابة للإشارات البصرية التي يتلقّاها من المُوجِّه عبر وسائط نقل الصور والفيديو. |
| 6 | الروبوتية؛ حتّى يتمكّن من استقبال الأشياء التي يتلقّاها من المُوجِّه عبر المنفذ ويعالجها. |

تغطي الإمكانات الموضحة بالأعلى جزءًا كبيرًا من مجال الذكاء الاصطناعي الواسع. سنستعرض هذه الإمكانات فيما يلي:

**معالجة اللغات الطبيعية** (NLP) هو أحد فروع الذكاء الاصطناعي الذي يَمنح أجهزة الحاسب القدرة على فهم الإنسان واللغة الطبيعية.

**تمثيل المعرفة** (Knowledge Representation) في الذكاء الاصطناعي يشير إلى عملية ترميز المعرفة البشرية في شكل مقروء آليًا لتتمكن الأنظمة المُستندة إلى الذكاء الاصطناعي من معالجتها واستخدامها. تأتي هذه المعرفة في صورٍ عدة تشمل: الحقائق، والقواعد، والمفاهيم، والعلاقات، والعمليات.

**الاستدلال المُؤتمت** (Automated Reasoning) يُشير إلى قدرة الأنظمة المُستنِدة إلى الذكاء الاصطناعي على استنتاج المعرفة الجديدة وتقديم الاستنتاجات المنطقية وفقًا لمجموعة من القواعد والفرضيات المُقدَمة.

**رؤية الحاسب** (Computer Vision) هي مجال الذكاء الاصطناعي الذي يُمكِّن الحاسب من تفسير وفهم المعلومات المرئية من العالم الحقيقي، مثل الصور ومقاطع الفيديو.

**الروبوتية** (Robotics) هي فرع الذكاء الاصطناعي الذي يُعنى بتصميم الروبوت، وبنائه، واستخدامه. ويتضمن الجمع بين التقنيات المتنوعة مثل: تعلُّم الآلة، ورؤية الحاسب، وأنظمة التحكّم لابتكار آلات ذكية ذاتية التحكُّم أو تتطلب الحد الأدنى من التوجيه البشري.

# الذكاء الاصطناعي: تاريخ مُمتَّد لتسعة عقود

## Artificial Intelligence: 9 Decades of History

بالرغم من أن عمر الذكاء الاصطناعي لا يتجاوز 100 عام، إلا أنه يتمتع بتاريخ غني يَمتد منذ الأربعينيات من القرن الماضي حتى اليوم. وفيما يلي استعراض للإنجازات البارزة في مجال الذكاء الاصطناعي في كل عِقد.

### الأربعينيات: البداية وأول خلية عصبية اصطناعية

**1943**: أُقترح النموذج الأول المبني على الخلايا العصبية الاصطناعية بحيث يمكن لكل خلية عصبية أن تكون في حالة نشطة (تشغيل) أو غير نشطة (إيقاف) وذلك وفق المحاكاة التي تتلقاها من الخلايا العصبية الأخرى المجاورة والمتصلة بها.

**1948**: في هذا العام ظهر روبوتان: **إلمر وإلسي** (Elmer and Elsie) وهما روبوتان ذاتيا التحكُّم، يمكنهما التنقل حول العقبات باستخدام الضوء واللمس.

### خمسينات القرن الماضي: نشأة الذكاء الاصطناعي

**1950**: ظهر اختبار تورنغ وهو اختبار يحدِّد قدرة الآلة على إظهار سلوك ذكي مكافئ لسلوك الإنسان أو يَصعُب تمييزه عنه. إلى جانب ظهور العديد من مفاهيم الذكاء الاصطناعي الرئيسة مثل: تعلُّم الآلة، والخوارزميات الجينية، والتعلُّم المعزَّز.

**1951**: صُمِّم **حاسب التعزيز التناظري العصبي العشوائي** (Stochastic Neural Analog Reinforcement Computer-SNARC) كأول حاسب يعمل بالشبكات العصبية.

**1958**: طُوِّرت **لغة ليسب** (Lisp)، وهي لغة برمجة مُصمَّمة خصيصًا للذكاء الاصطناعي. وفي العام نفسه، نُشرت ورقة بحثية حول **متلقي المشورة الافتراضي** (Hypothetical Advice Taker)، وهو نظام الذكاء الاصطناعي القادر على التعلُّم من التجربة تمامًا مثل البشر.

### الستينيات والسبعينيات من القرن الماضي: أول شتاء للذكاء الاصطناعي

**1964**: ظهر برنامج **إليزا** (ELIZA) وهو أول برنامج لمعالجة اللغات الطبيعية وهي الأصل الذي تفرّع منه جميع روبوتات الدردشة اليوم.

**1974-1980**: تُعرف هذه الفترة باسم أول شتاء للذكاء الاصطناعي. حيث انخفض تمويل مشروعات الذكاء الاصطناعي في هذه الفترة نظرًا لقلة التقدم المُحرز في هذا المجال، وانخفاض تأثيره في تطبيقات الحياة اليومية. أحد الانتقادات الرئيسة كانت عدم قدرة تقنيات الذكاء الاصطناعي على معالجة مشكلة الانفجار التوافقي التي جعلت قابلية تطبيقها محدودة على بعض المشكلات ومجموعات البيانات الصغيرة للغاية.

### الثمانينيات والتسعينيات من القرن الماضي وثاني شتاء للذكاء الاصطناعي

**1980**: أُطلق أول نظام خبير تجاري ناجح مُصمَّم لمحاكاة القدرة على صُنع القرار مثل الإنسان.

**1987-1993**: تُعرف هذه الفترة باسم ثاني شتاء للذكاء الاصطناعي. فطبيعة أنظمة الذكاء الاصطناعي في المراحل المُبكرة كانت مستندة على القواعد، والتي بدورها قيدت من قابليتها للتطبيق وجعلتها غير قادرة على حل مشاكل الحياة الواقعية الرئيسة.

**1997**: تحقق الفوز الأول لبرنامج الذكاء الاصطناعي على بطل العالم في الشطرنج، حيث نجح الحاسب العملاق **ديب بلو** (Deep Blue) في هزيمة بطل العالم في الشطرنج **جاري كاسبارو** (Gary Kasparov).

### الألفينيات: فترة الانتشار واسع النِّطاق، والدعم الكبير للمكوّنات المادية والبرمجية، وتطورها

**2005**: طوَّرت **جامعة ستانفورد** (Stanford University) السيارة ذاتية القيادة **ستانلي** (STANLEY) التي فازت في تحدي السيارات ذاتية القيادة. كما بدأ الجيش الأمريكي الاستثمار في الروبوتات ذاتية التحكّم.

**2009**: استُخدمت **وحدات معالجة الرسومات** (Graphics Processing Units – GPUs) لتدريب الشبكات العصبية للتعلُّم العميق للمرة الأولى. أدى استخدام المكونات المادية المتخصصة إلى تسارع وتيرة تدريب الشبكات المُعقَّدة على مجموعات كبيرة جدًا من البيانات، مما أدى بدوره إلى عصرٍ جديد من التعلُّم العميق والذكاء الاصطناعي.

### العقدين الثاني والثالث من القرن الحادي والعشرين: العصر الذهبي

**2011**: هزم نظام الإجابة على الأسئلة المعروف باسم **واتسون** (Watson) أفضل لاعبين في العالم في برنامج المسابقات الأمريكي **جيوباردي** (Jeopardy)، حيث تمكّن واتسون من فهم الأسئلة والإجابة عليها بنجاح، مما شكّل طفرة في استخدام الذكاء الاصطناعي لِفهم اللغة الطبيعية.

**2012**: ظهر نظام الذكاء الاصطناعي الذي يُترجِم فوريًا اللغة الإنجليزية المنطوقة إلى اللغة الصينية المنطوقة.

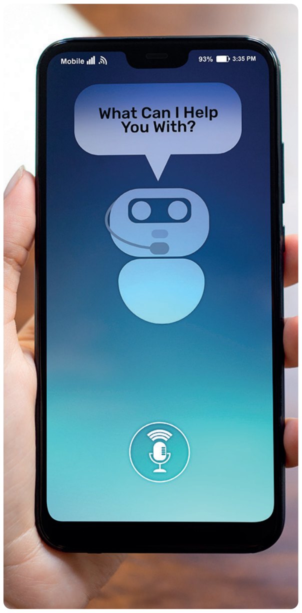
**2021**: ظهر نظام القيادة الذاتية الكامل الذي يَستخدِم الشبكات العصبية المُدرَّبة على سلوك مئات الآلاف من السائقين.

**2022**: ظهر **روبوت دردشة المُحوِّل التوليدي مُسبق التدريب** (Generative Pre-trained Transformer - ChatGPT) وهو روبوت الدردشة المبني على مجموعة كبيرة من النماذج اللغوية. هذه النماذج مُهيأة بدقة باستخدام كلٍ من تقنيات التعلُّم المُوجّه والمُعزّز لمحاكاة المحادثات البشرية.

# تطبيقات الذكاء الاصطناعي Applications of AI

الذكاء الاصطناعي هو تقنية سريعة التطور لديها القدرة على تحوُّل مجموعة واسعة من القطاعات والصناعات. في هذه الوحدة ستستكشف تطبيقات الذكاء الاصطناعي المتنوعة، وكيفية استخدامها في إجراء تحسينات وابتكارات في مجموعة متنوعة من القطاعات والصناعات.

## المساعدون الافتراضيون Virtual Assistants

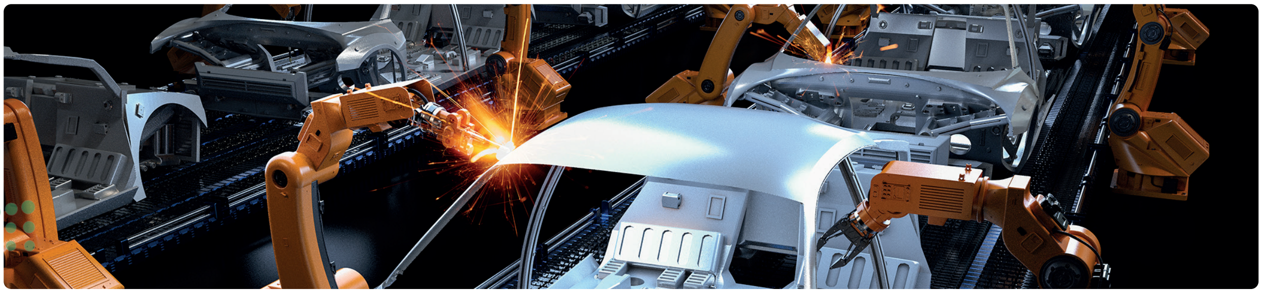


شكل 1.3: المحادثة مع روبوت الدردشة

واحدة من أشهر تطبيقات الذكاء الاصطناعي هي تطبيقات المساعدين الافتراضيين الذين يمكنهم التواصل مع المُستخدِمين عبر التفاعلات النصيّة أو الصوتية، ويمكن الوصول إليهم عبر الأجهزة المادية مثل: الهواتف الذكية، والأجهزة اللوحية، أو مكبرات الصوت الذكية، ويمكن استخدامهم لأداء مجموعة واسعة من المهام مثل: إعداد التذكيرات، والإجابة على الأسئلة، وتشغيل الوسائط الصوتية، وطلب المنتجات أو الخدمات. أحد الأمثلة الأكثر شهرة على تطبيقات الذكاء الاصطناعي في هذا المجال هو **سيري** (Siri) من شركة **آبل** (Apple). وهناك شركات أخرى طوّرت مساعدين افتراضيين: مثل **أليكسا** (Alexa) التابع لشركة **أمازون** (Amazon)، و**المساعد الافتراضي لقوقل** (Google's Assistant)، و**كورتانا** (Cortana) التابع لشركة **مايكروسوفت** (Microsoft). وبمرور الوقت تطوَّرت قدرة هذه التطبيقات على الفهم والاستجابة لعدد متزايد من الأوامر والاستفسارات والرد عليها. على سبيل المثال، يمكن استخدام المساعد الافتراضي للتحكم في الأجهزة المنزلية الذكية مثل: التحكم في درجة الحرارة، والإضاءة، والأجهزة الكهربائية. وقد يتمثل المساعد الافتراضي في صورة روبوتات الدردشة المتخصصة المُصمَّمة عادةً لتقديم المعلومات والإجابة على الأسئلة في مجال محدَّد، على سبيل المثال، في تطبيقات خدمة العملاء تُستخدَم روبوتات الدردشة المبنية على تقنية الذكاء الاصطناعي في الإجابة على أسئلة العملاء حول المنتجات أو الخدمات، وتحديد المشكلات وعلاجها، وتقديم المعلومات حول طلباتهم وحساباتهم. يمكن الوصول إلى روبوتات الدردشة عبر مجموعة واسعة من القنوات مثل: مواقع الويب، وتطبيقات المراسلة، ووسائل التواصل الاجتماعي، ويمكنها تقديم خدمات المساعدة على مدار الساعة طوال أيام الأسبوع. يمكنك الاطلاع على مثال لأحد تطبيقات روبوت الدردشة في الشكل 1.3.

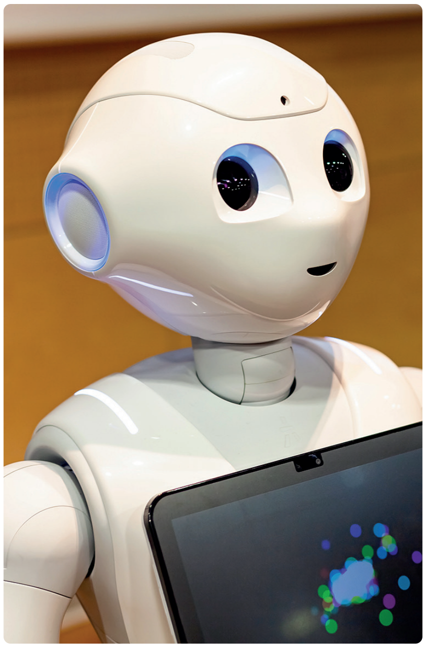
## الروبوتية Robotics

ارتبط الذكاء الاصطناعي منذ بداياته بالروبوتية، فإذا كان الروبوت هو التصوير المادي للكائن الاصطناعي، فإنّ الذّكاء الاصطناعي يمثل دماغ الروبوت، ويَمنحه القدرة على الشعور بالبيئة من حوله، واتخاذ القرارات، والتكيف مع الظروف المتغيرة. كما يمكن للروبوتات الذكية تطبيق هذه الإمكانات والقدرات لأداء مجموعة واسعة من المهام دون التدخل البشري، مثل: مهام التصنيع، والاستكشاف، والبحث والإنقاذ، والعديد من المهام الأخرى. الشكل 1.4 يوضِّح خط تجميع روبوتي في مصنع سيارات.


شكل 1.4: خط تجميع روبوتي في مصنع سيارات


وزارة التعليم
Ministry of Education
2025 - 1447

إنّ أحد أقدم الأمثلة على تطبيق الذكاء الاصطناعي في الروبوتية هو تطوير روبوتات المصانع المُستخدَمة في أداء المهام مثل: اللحام، والدهانات، والتجميع. منذ ذلك الحين، تطوَّر استخدام الذكاء الاصطناعي في الروبوتية إلى حد كبير، مع تطور الخوارزميات المتقدمة واستخدام تعلُّم الآلة لتحسين أداء الروبوت. وكانت إحدى الإنجازات البارزة في استخدام الذكاء الاصطناعي في الروبوتية تطوير الروبوتات البشرية، مثل: روبوت **هوندا أسيمو** (Honda's ASIMO) وقد سُمّي بذلك اختصارًا لمفهوم **الخطوة المتقدمة في النقل الإبداعي** (Advanced Step in Innovative Mobility) والذي قُدّم للمرة الأولى في عام 2000 وكان قادرًا على السير وأداء المهام الأساسية.



شكل 1.5: الروبوت بيبر

## الروبوتات الشبيهة بالبشر Humanlike Robots

طوّرت شركة **الدبران روبوتكس** (Aldebaran Robotics) الروبوتان الشبيهان بالبشر **بيبر** (Pepper) و**ناو** (Nao)، اللّذانِ صُمِّما لأغراض البحث والتطوير في مجال التفاعل بين الإنسان والروبوت، وقد استُخدما على نطاق واسع في مجالات البحث، والتعليم، والترفيه. أمّا **بيبر** (Pepper) فهو روبوت اجتماعي مُصمَّم للتفاعل مع الأشخاص بصورة طبيعية باستخدام كاميرا، وميكروفونات، ومُستشعرات اللمس لإدراك البيئة من حوله، والاستجابة لتصرفات وعواطف الأشخاص من حوله. يتمتع هذا الروبوت بالعديد من الخصائص التي تسمح له بالتعرّف على الوجوه، وفهم الكلام، والاستجابة للإيماءات. الشكل 1.5 يعرض صورة للروبوت بيبر. أمّا **ناو** (Nao) فهو روبوت مُدمج أصغر حجمًا مُصمَّم للتفاعل مع البشر، ويحتوي هذا الروبوت مثل السابق على مجموعة من المُستشعرات التي تسمح له بإدراك البيئة من حوله، إلى جانب الكاميرات، والميكروفونات للتعرّف على الكلام والوجوه. ويمتاز هذا الروبوت بأنّه قابل للتخصيص والبرمجة بدرجة توافقية عالية، مما يجعله الخيار الأمثل للباحثين والدارسين الذين يرغبون في دراسة وتطوير تطبيقات جديدة للروبوتات الشبيهة بالبشر.

**في عام 2017 كانت الروبوت صوفيا (Sophia) أول روبوت يحصل على الجنسية السعودية، وفي عام 2023 طورت المملكة العربية السعودية سارة (Sarah)، وهي الروبوت التفاعلي الأول من نوعه.**

## السيارات ذاتية القيادة Self-Driving Cars

كان الإنجاز المهم الآخر هو تطوير السيارات ذاتية القيادة كما في الشكل 1.6 وهي سيارات تَستخدِم الذكاء الاصطناعي للانتقال عبر الطرق واتخاذ القرارات حول كيفية التفاعل الآمن مع المركبات الأخرى ومع المُشاة. أحد المتطلبات الرئيسة لهذه التطبيقات هو القدرة على معالجة البيانات المرئية مثل الصور ومقاطع الفيديو وفهمها، ويشار إلى ذلك عادة باسم **رؤية الحاسب** (Computer Vision)، ويمكن استخدام خوارزميات رؤية الحاسب للتعرُّف على الكائنات، والأشخاص، والخصائص الأخرى في الصور ومقاطع الفيديو، إلى جانب فهم سياق المحتوى ومعناه. ولهذا المجال العديد من التطبيقات غير الروبوتية مثل: التعرُّف على الوجه، وإدارة المحتوى، وتحليل الوسائط. وكان أحد الإنجازات البارزة في استخدام الذكاء الاصطناعي في تحليل الصور ومقاطع الفيديو تطوير خوارزميات التعلُّم العميق، التي يُمكنها تحليل كميات كبيرة من البيانات وتحديد الأنماط المُعقَّدة في الصور ومقاطع الفيديو.

شكل 1.6: سيارة ذاتية القيادة

# المجالات التي تأثّرت بالذكاء الاصطناعي Industries Affected by AI

## التعليم Education

على مدى العقود القليلة الماضية، كانت هناك العديد من الإنجازات الرئيسة لاستخدام الذكاء الاصطناعي في التعليم. بما في ذلك تطوير أنظمة التدريس القائمة على الذكاء الاصطناعي التي تَستخدِم تقنيات معالجة اللغات الطبيعية للتفاعل مع الطلبة وتقديم الملاحظات حول أعمالهم. ثم ظهرت منصّات التعلُّم التكيُّفي التي تَستخدِم خوارزميات تعلُّم الآلة لتخصيص العملية التعليمية لكل طالب استنادًا إلى نقاط قوته وضعفه. بعدها، طُوِّرت أنظمة التصحيح القائمة على الذكاء الاصطناعي التي تَستخدِم خوارزميات معالجة اللغات الطبيعية وتعلُّم الآلة لتصحيح الواجبات المكتوبة وتقديم الملاحظات. وفي الآونة الأخيرة، حدث دمج بين المساعدين الافتراضيين وروبوتات الدردشة في مجال التعليم لتقديم الدعم المخصص للطلبة والإجابة على أسئلتهم بشكل فوري. يمكن استخدام الذكاء الاصطناعي لتحليل البيانات حول أداء الطلبة، وخياراتهم المفضلة في التعليم، وغيرها من العوامل الأخرى اللازمة لوضع خطط تعليمية مخصصة للطلبة، وتقديم التوصيات بشأن المواد أو الأنشطة التي من المرجح أن تفيدهم بفعالية.

**مزايا الذكاء الاصطناعي في التعليم**

**AI benefits in education**

- يوفّر وقت المُعلّمين والأساتذة الجامعيين.
- يُمكن لمُعلّمي الذكاء الاصطناعي (AI Tutors) مساعدة الطلبة.
- يساعد المُعلّم على أن يصبح معلّمًا محفّزًا.
- تُقدّم الوظائف المُستنِدة على الذكاء الاصطناعي الملاحظات لكل من الطلبة والمُعلّمين.

## الرعاية الصحية Healthcare

الرعاية الصحية هي مجال آخر حقّق تقدمًا كبيرًا بفضل الذكاء الاصطناعي. كانت الابتكارات الأولى في صورة الأنظمة التشخيصية القائمة على الذكاء الاصطناعي واستخدامه في اكتشاف الأدوية. ثم دمجه مع السجلات الصحية الإلكترونية لاستخراج المعلومات ذات الصلة، وفي العِقد الثاني من القرن الحادي والعشرين، طُوِّرت أنظمة التطبيب عن بُعد القائمة على الذكاء الاصطناعي. واليوم، يُساعد الذكاء الاصطناعي الحديث في إنشاء خطط علاجية مُخصصة للمريض، واستخدام أجهزة تقنية يرتديها لمتابعة حالته الصحية. ويلعب الذكاء الاصطناعي دورًا كبيرًا في مجال الرعاية الصحية، فهو يُمكِّن الأطباء ومقدّمي خدمات الرعاية الصحية الآخرين من تحليل كميات كبيرة من البيانات واتخاذ القرارات حول رعاية المرضى. قد تأتي البيانات من مصادر متنوعة مثل: السجلات الطبية، والفحوصات المعملية، وكذلك الصور مثل: الأشعة السينية أو الأشعة المقطعية، كما تُستخدَم خوارزميات رؤية الحاسب الحديثة بصورة متكررة للكشف عن التشوهات والمساعدة في التشخيص الطبي.

شكل 1.7: تحليل البيانات الصحية

## الزراعة والنمذجة المُناخية Agriculture and Climate Modeling

يُستخدَم الذكاء الاصطناعي في الزراعة لتحسين إنتاج المحاصيل الزراعية ورفع كفاءة الممارسات الزراعية. ويتحقق ذلك بالتحليل المستمر للبيانات حول حالة التربة، وأنماط الطقس، والعوامل الأخرى للتنبؤ بأفضل وقت لزراعة المحاصيل الزراعية وريِّها وحصادها. كما يُستخدَم الذكاء الاصطناعي في مراقبة المحاصيل طوال الوقت وتحديد المشكلات التي قد تصيبها مثل: الآفات أو الأمراض، مما يسمح للمزارعين باتخاذ اللازم قَبل أن تؤثر تلك المشكلات على جودة المحاصيل الزراعية، وأحد الأمثلة المُبتكرة على تطبيقات الذكاء الاصطناعي في الزراعة هو استخدام خوارزمية صُنع القرارات البسيطة لتحسين مواعيد الري. ومن الإنجازات الرئيسة الأخرى استخدام شبكات المُستشعرات لمراقبة المحاصيل الزراعية، ومعايرة التطبيقات العلاجية الرئيسة مثل الأسمدة والمبيدات. وفي الآونة الأخيرة، استُخدِمت الصور المُلتقطة بالطائرات المُسيَّرة والأقمار الصناعية لتحليل المحاصيل الزراعية على نطاق واسع، كما في الشكل 1.8 الذي يعرض طائرة مُسيَّرة تُستخدَم لتسميد أحد الحقول.

شكل 1.8: التسميد باستخدام الطائرات المُسيَّرة

أمّا النمذجة المُناخية فهي مجال آخر يرتبط ارتباطًا وثيقًا بالزراعة، وقد تأثر كثيرًا بالذّكاء الاصطناعي الذي بدأت تطبيقاته في هذا المجال في وقت مُبكر، مع تطوير أنظمة التنبؤ بالطقس القائمة عليه. ولاحقًا، استُخدِم الذكاء الاصطناعي لتحليل كميات كبيرة من البيانات حول التغيرات المُناخية والتنبؤ بالأنماط المستقبلية، وتأتي هذه البيانات من مصادر متنوعة، بما في ذلك صور الأقمار الصناعية، وملاحظات محطات الطقس، والمحاكاة الحاسوبية. واليوم، يُستخدَم الذكاء الاصطناعي في مجموعة واسعة من تطبيقات النمذجة المُناخية مثل: التنبؤ بآثار التغيرات المُناخية على مناطق محدَّدة، وتحليل وفهم أسباب الظواهر الجوية المتطرفة وفهمها، ووضع الاستراتيجيات الفعّالة للتخفيف من التغيرات المُناخية أو التكيّف معها.


وزارة التعليم
Ministry of Education
2025 - 1447

## الطاقة Energy

أثَّر الذكاء الاصطناعي كثيرًا على مجال الطاقة، وذلك عن طريق تمكين الشّركات من ترشيد استخدامها وتقليل الهَدْر، وتحسين الكفاءة. أحد الأمثلة على ذلك استخدام خوارزميات تعلُّم الآلة لتحليل البيانات حول استخدامات الطاقة وتحديد طرائق تقليل الهَدْر وترشيد الاستهلاك. في التسعينيات من القرن الماضي، استُخدِم الذكاء الاصطناعي للتنبؤ بموارد الطاقة المُتجددة وتحسين استخدامها. وكان تطورًا رئيسًا مكَّن شركات الطاقة من التخطيط بصورة أفضل لدمج موارد الطاقة المتجددة في عملياتها.

شكل 1.9: الطاقة الكهربائية النظيفة من الألواح الكهروضوئية الشمسية

شهد العِقد الأول من القرن الحادي والعشرين دمج الذكاء الاصطناعي في الشبكات الذكية، التي تستخدِم خوارزميات تعلُّم الآلة في تحليل البيانات حول استخدام الطاقة وضبط العرض والطلب طوال الوقت، حيث ساهم ذلك في تحسين كفاءة توزيع الطاقة والحد من الهدر، وفي العِقد الثاني من القرن الحادي والعشرين، استُخدِم الذكاء الاصطناعي لتطوير أنظمة تخزين الطاقة التي يمكنها تخزين الطاقة الزائدة واستخدامها عند الحاجة. وكان تطورًا رئيسًا مكَّن شركات الطاقة من إدارة الاستخدام المُتقطِّع بشكل أفضل لموارد الطاقة المتجددة مثل: الطاقة الشمسية وطاقة الرياح. يعرض الشكل 1.9 الألواح الكهروضوئية الشمسية، وفي السنوات الأخيرة، استُخدِم الذكاء الاصطناعي لزيادة كفاءة استخدام الطاقة بتحليل البيانات حول استخدام الطاقة وتحديد طرائق الحد من الهدر، وشمل ذلك تطوير الأنظمة المُستنِدة على الذكاء الاصطناعي التي تُستخدَم في تحسين استخدام الطاقة في المباني، والمصانع، ومن قِبل كبار مُستهلكي الطاقة. كما استُخدِم الذكاء الاصطناعي في صناعة النفط والغاز لتحليل البيانات حول الحفر والإنتاج وتحسين العمليات.

## تطبيق القانون Law Enforcement

يُستخدَم الذكاء الاصطناعي بكثافة في مجال تطبيق القانون للتنبؤ بالجرائم والحيلولة دون وقوعها. وعلى وجه التحديد، يُستخدَم الذكاء الاصطناعي لتحليل البيانات من مصادر مختلفة، مثل: سجلات الجرائم، ووسائل التواصل الاجتماعي، وكاميرات المراقبة لتحديد أنماط وتوجُّهات الأنشطة الإجرامية والتنبؤ بها. على سبيل المثال طُوِّر الذكاء الاصطناعي في التعرُّف على الوجوه (شكل 1.10). ولاحقًا، دُمِج في أنظمة إرسال قوات الشرطة واستُخدِم لمراقبة منصّات وسائل التواصل الاجتماعي بحثًا عن التهديدات المحتملة. وفي الآونة الأخيرة، استُخدِم الذكاء الاصطناعي لتطوير طائرات مُسيَّرة لمراقبة وتحليل تسجيلات الفيديو من الكاميرات التي يرتديها ضباط تطبيق القانون. كما لعِب الذكاء الاصطناعي دورًا كبيرًا في تمكين الجهات المسؤولة من تحليل كميات كبيرة من البيانات، وتحديد الأنماط والتوجهات، واتخاذ القرارات المُستنيرة حول كيفية منع الجريمة والتصدي لها.

شكل 1.10: تقنيات التعرُّف على الوجه وتحديد الهوية الشخصية

وزارة التعليم
Ministry of Education
2025 - 1447

# تمرينات

1

| حدِّد الجملة الصحيحة والجملة الخاطئة فيما يلي: | صحيحة | خاطئة |
|---|---|---|
| 1. وضع علماء الرياضيات الأُسس لفهم الحوسبة والمنطق حول الخوارزميات. | ○ | ○ |
| 2. يُحدِّد اختبار تورنغ ما إذا كان الحاسب يتمتع بسلوك شبيه بالإنسان أم لا. | ○ | ○ |
| 3. كان إلمر (Elmer) وإلسي (Elsie) أول روبوتين يتنقلان حول العقبات باستخدام الضوء واللمس. | ○ | ○ |
| 4. استُخدِم الذكاء الاصطناعي فقط في الروبوتات المُستخدَمة في الصناعات التحويلية. | ○ | ○ |
| 5. لم يكن للذكاء الاصطناعي أي تأثير يُذكر في مجال الطاقة. | ○ | ○ |

2 ما الذكاء الاصطناعي (AI)؟

3 اشرح بإيجاز بعض تطبيقات الذكاء الاصطناعي المُستخدَمة في الحياة اليومية.

4 وضِّح بعض الأحداث التاريخية الرئيسة التي أثَّرت في تطور الذكاء الاصطناعي في الأربعينيات والخمسينيات من القرن الماضي.

5 اشرح كيف استُخدمت التطبيقات التجارية تقنيات الذكاء الاصطناعي للمرة الأولى في العِقد الثاني من القرن الحادي والعشرين.

6 لخِّص كيفية استخدام تطبيقات الذكاء الاصطناعي في التصدي لتغيرات المُناخ عبر النمذجة المُناخية والتحسينات في مجال الطاقة.

وزارة التعليم
Ministry of Education
2025 - 1447

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الثاني

# هياكل البيانات في الذكاء الاصطناعي

## أهمية هياكل البيانات في الذكاء الاصطناعي

## The Importance of Data Structures in AI

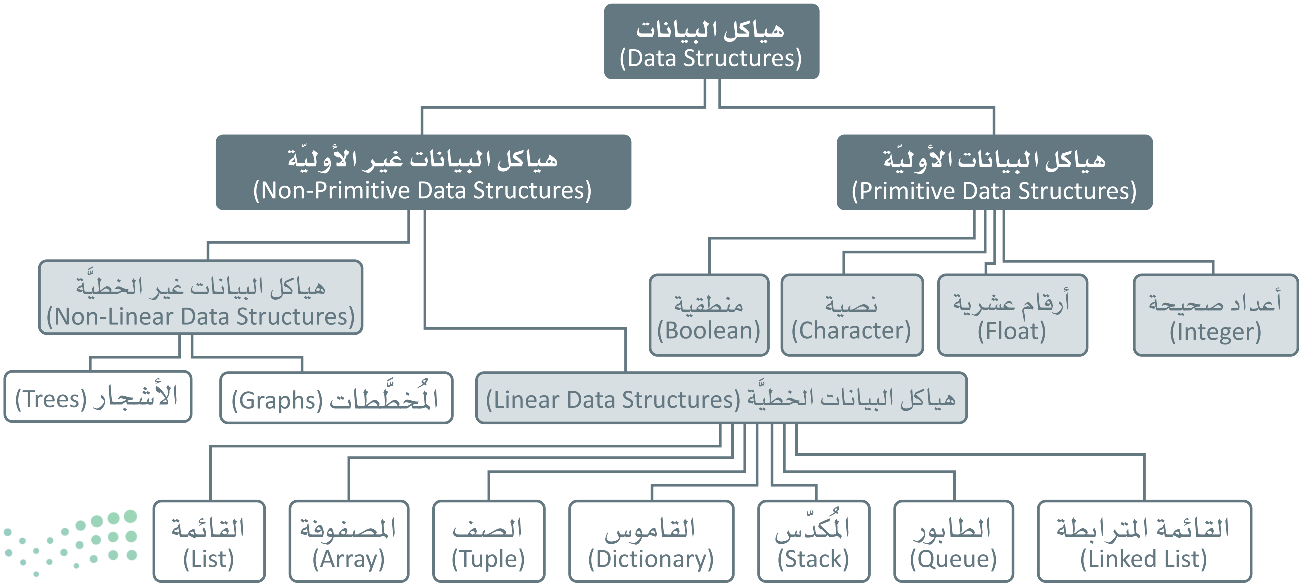
**هياكل البيانات (Data Structure):**
هياكل البيانات هي تقنية لتخزين وتنظيم البيانات في الذاكرة لاستخدامها بكفاءة.

للبيانات أهمية كبرى في مجالات الذكاء الاصطناعي؛ لأنها الأساس المُستخدَم في تدريب نماذج تعلُّم الآلة، حيث تُحدِّد جودة البيانات وكميّتها المتوافرة دقة وفعالية نماذج الذكاء الاصطناعي. ودون بيانات كافية وذات صلة، لن تتعلّم خوارزميات الذكاء الاصطناعي الأنماط، ولن تقوم بالتنبؤات، ولن تتمكن من أداء المهام بفاعلية. وبالتالي، تلعب البيانات دورًا رئيسًا في تشكيل قدرات وإمكانات صُنع القرار لدى أنظمة الذكاء الاصطناعي. هياكل البيانات لها أهمية كبيرة في الذكاء الاصطناعي؛ لأنها توفر طريقة فعَّالة لتنظيم وتخزين البيانات، كما تسمح باسترجاع ومعالجة البيانات بكفاءة. وكذلك، تُحدِّد مدى تعقيد وكفاءة الخوارزميات المُستخدَمة في معالجة البيانات، وبالتالي تؤثر مباشرةً على أداء أنظمة الذكاء الاصطناعي. على سبيل المثال، يمكن تحسين سرعة وقابلية خوارزميات الذكاء الاصطناعي للتوسُّع باستخدام هياكل البيانات المناسبة، مما يجعلها أكثر ملاءمة للتطبيقات في العالم الحقيقي. وكذلك، تساعد هياكل البيانات المُصمَّمة جيدًا في تقليل استخدام الذاكرة وزيادة كفاءة الخوارزميات، لتمكينها من معالجة مجموعات أكبر من البيانات. تُخزِّن أجهزة الحاسب البيانات وتُعالجها بسرعة ودقة فائقتين. لذلك، من الضروري تخزين البيانات بكفاءة، وتوفير الوصول إليها بطريقة سريعة. يمكن تصنيف هياكل البيانات على النحو التالي:

- هياكل البيانات الأوليّة.
- هياكل البيانات غير الأوليّة.

يُطلَق على البيانات البسيطة كذلك البيانات الأوليّة، أو الخام، أو الأساسية.

يوضِّح المُخطَّط في الشكل 1.11 تصنيف هياكل البيانات.




شكل 1.11: مُخطَّط هياكل البيانات


وزارة التعليم
Ministry of Education
2025 - 1447

## هياكل البيانات الأوليّة Primitive Data Structures

يُشار إلى هياكل البيانات الأوليّة باسم هياكل البيانات الأساسية في لغة البايثون، ويحتوي هذا النوع من الهياكل على قيم بسيطة للبيانات. تُخبر أنواع البيانات البسيطة المُترجِّم بنوع البيانات التي يُخزِّنها. هياكل البيانات الأوليّة في لغة البايثون هي:

- **الأرقام** (Numbers): تُستخدَم الأرقام لتمثيل البيانات الرقمية وهي:
  - الأعداد الصحيحة
  - الأرقام العشرية
- **السلاسل النصيّة** (Strings): السلاسل النصيّة هي مجموعات من الأحرف والكلمات.
- **البيانات المنطقية** (Boolean): تكون قيم البيانات المنطقية إما صحيحة أو خاطئة.

تُستخدَم أنواع مختلفة من هياكل البيانات لتطبيقات الحاسب ومهامّه المختلفة بناءً على ما يتطلبه المشروع والقيود المفروضة على الذاكرة.

## هياكل البيانات غير الأوليّة Non-Primitive Data Structures

هياكل البيانات غير الأوليّة هي هياكل متخصصة تُخزِّن مجموعة من القيم. يكتبها المُبرمِّج ولا تُعرَّف بلغة البايثون مثل هياكل البيانات الأوليّة.

يمكن تقسيم هياكل البيانات غير الأوليّة كذلك إلى نوعين:

- **هياكل البيانات الخطيَّة أو المُتسلسلة** (Linear or Sequential Data Structures): تُخزِّن هياكل البيانات الخطيّة عناصر البيانات في تسلسل معيّن.
- **هياكل البيانات غير الخطيَّة** (Non-linear Data Structures): لا تحتوي هياكل البيانات غير الخطيَّة على ارتباط تسلسلي بين عناصر البيانات، ويُمكن ربط أي زوج أو مجموعة من عناصر البيانات معًا، والوصول إليها دون تسلسل مُحدَّد.

### هياكل البيانات الخطيَّة Linear Data Structures

تُخزِّن هياكل البيانات الخطيَّة عناصر البيانات في تسلسل معيّن. في هذا الدرس ستتعلّم بعض هياكل البيانات الخطيّة مثل: **المُكدّس** (Stack) **والطابور** (Queue)، وهما نوعان من هياكل البيانات الأكثر استخدامًا في الحياة اليومية.

شكل 1.12: كومة من الكتب كمثال واقعي على المُكدّس

#### المُكدّس Stack

يمكن تمثيل المُكدّس في الواقع بمجموعة من الكتب رُصَّت فوق بعضها البعض، كما هو موضّح في الشكل 1.12. فلإنشاء تلك المجموعة، عليك أن تضع الكتب بعضها فوق بعض، وعندما تريد استخدام أحد الكتب، عليك أخذ الكتاب من أعلى المجموعة. وللوصول إلى الكتب الأخرى عليك إنزال الكتب من أعلى المجموعة.

قد يكون حجم المُكدّس ثابتًا أو متغيّرًا ديناميكيًا. تُطبِّق لغة البايثون المُكدّسات باستخدام القوائم.

**قاعدة المُضاف آخرًا يَخرُج أولًا (Last In First Out (LIFO) Rule):**

آخر عنصر مُضاف يمكن الوصول إليه أولًا.



وزارة التعليم
Ministry of Education
2025 - 1447

## العمليات في المُكدّس Operations on the stack

هناك عمليتان رئيستان في المُكدّس:

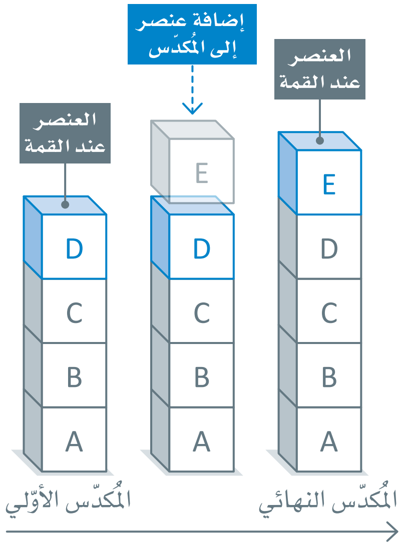
- **إضافة عنصر** (Push): تُستخدَم العملية لإضافة عنصر في قمة المُكدّس.
- **حذف عنصر** (Pop): تُستخدَم العملية لحذف عنصر من قمة المُكدّس.

### عملية إضافة عنصر Push operation

يُطلَق على عملية إضافة عنصر جديد إلى المُكدّس اسم **إضافة عنصر** (Push).

يَستخدِم المُكدّس مؤشّرًا يُطلق عليه مؤشر **الأعلى** (Top)، ويُشير إلى العنصر الموجود في قمة المُكدّس، وعند إضافة عنصر جديد إلى المُكدّس:

- تزداد قيمة مؤشر الأعلى بقيمة واحدة لإظهار الموقع الجديد الذي سيُضاف العنصر فيه.
- يُضاف العنصر الجديد إلى قمة المُكدّس.




شكل 1.13: عملية إضافة عنصر إلى المُكدّس

#### فَيْض المُكدّس Stack Overflow

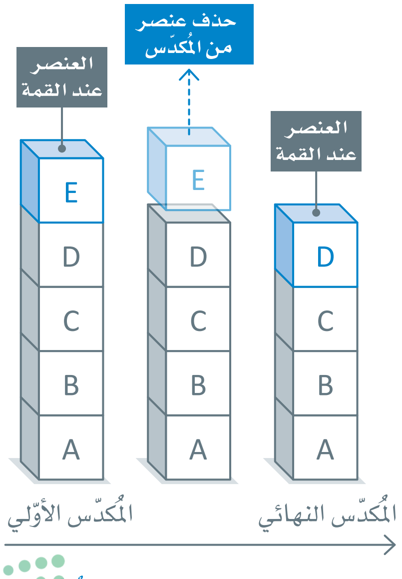
يتميز المُكدّس بسعة تخزينية مُحدَّدة تعتمد على ذاكرة الحاسب. إذا كانت الذاكرة ممتلئة، فإن إضافة عنصر جديد سينتج عنها مشكلة **فَيْض المُكدّس** (Stack Overflow). ويقصد بها تجاوز السعة؛ لذا يجب التحقق من امتلاء ذاكرة المُكدّس قبل إضافة أي عنصر جديد.

### عملية حذف عنصر Pop operation

يُطلق على عملية حذف عنصر من المُكدّس اسم **حذف عنصر** (Pop).

عند حذف عنصر من المُكدّس:

- يُحذَف العنصر من قمة المُكدّس.
- تنخفض قيمة مؤشر الأعلى بقيمة واحد لإظهار العنصر التالي عند قمة المُكدّس.




شكل 1.14: عملية حذف عنصر من المُكدّس

#### غَيْض المُكدّس Stack Underflow

إذا كنت ترغب في حذف عنصر من المُكدّس، عليك التّحقّق أولًا من أن المُكدّس يحتوي على عنصر واحد على الأقل؛ فإذا كان المُكدّس فارغًا، سينتج عن ذلك مشكلة **غَيْض المُكدّس** (Stack Underflow) ويقصد بها الانخفاض عن الحد الأدنى للسعة.

# المُكدّس في لغة البايثون Stack in Python

تُمثَّل المُكدّسات في لغة البايثون باستخدام القوائم التي بدورها تُقدِّم بعض العمليات التي يُمكن تطبيقها مباشرةً على المُكدّسات.

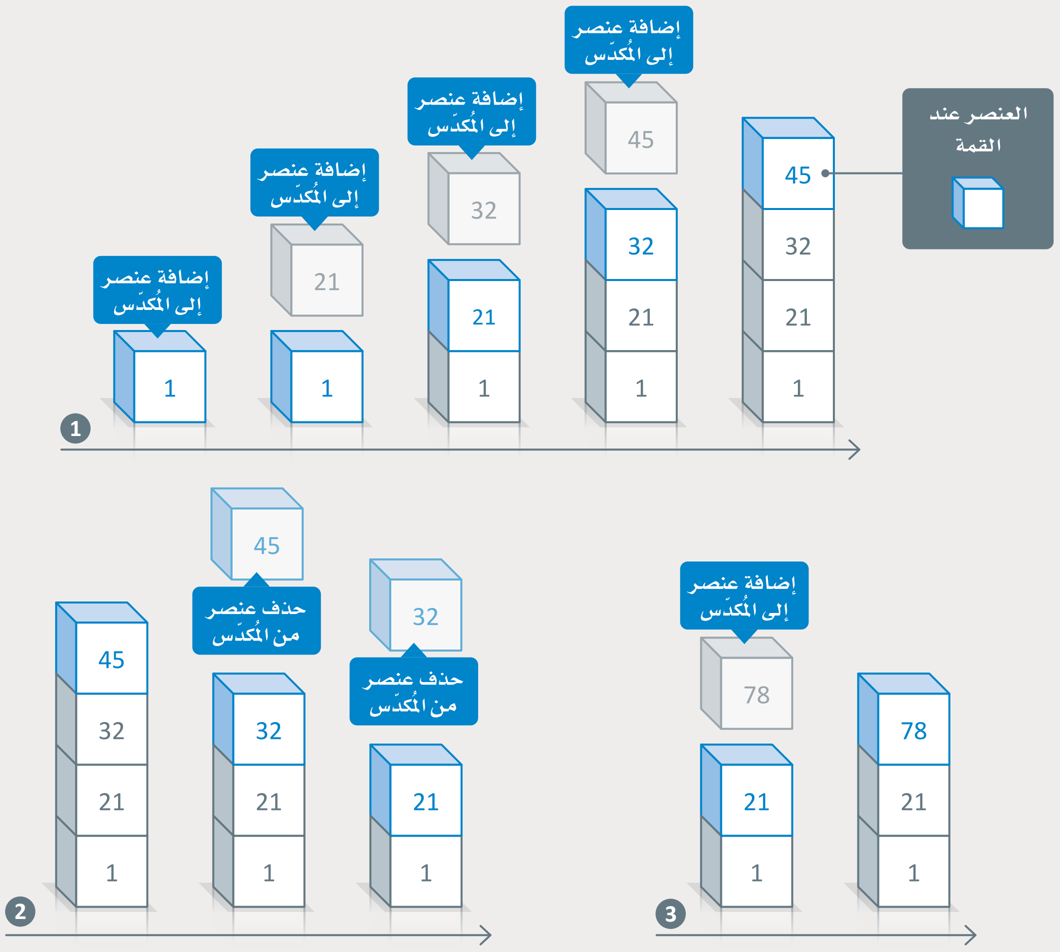
**جدول 1.2: طرائق المُكدّس**

| الطريقة | الوصف |
|---|---|
| listName.append(x) | إضافة العنصر x إلى نهاية القائمة. |
| listName.pop( ) | حذف العنصر الأخير من القائمة. |

تُطبَّق عملية إضافة عنصر للمُكدّس في لغة البايثون باستخدام دالة **append**.

ستشاهد مثالًا على تطبيق المُكدّس في لغة البايثون:

1. أنشئ المُكدّس لتخزين مجموعة من الأرقام (1، 21، 32، 45).
2. استخدِم عملية **حذف عنصر** (Pop) من المُكدّس مرتين لحذف العنصرين الأخيرين (32، 45) من المُكدّس.
3. استخدِم عملية **إضافة عنصر** (Push) إلى المُكدّس لإضافة عنصر جديد (78) إلى المُكدّس.




شكل 1.15: مثال على المُكدّس


وزارة التعليم
Ministry of Education
2025 - 1447

## مفكرة جوبيتر Jupyter Notebook

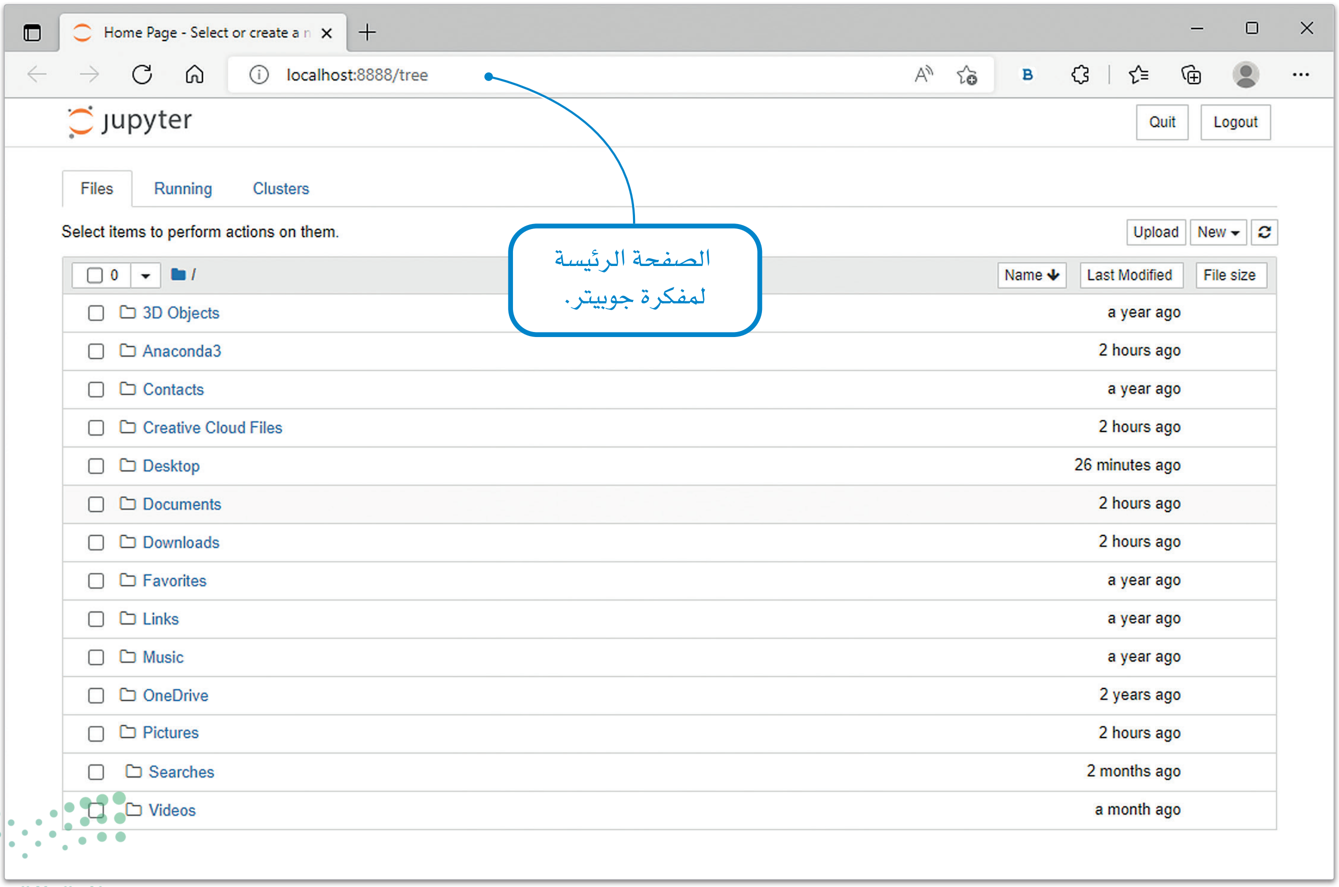
في هذه الوحدة ستكتب برنامجًا بلغة البايثون باستخدام **مفكرة جوبيتر** (Jupyter Notebook). وهي تطبيق الويب المُستخدَم لإنشاء المُستندات الحاسوبية ومشاركتها. كل مُستند يُسمى مفكرة، ويحتوي على المقطع البرمجي الذي كتبتَه، والتعليقات، والبيانات الأوليّة والمُعالجة، وتصوّرات البيانات. ستَستخدِم الإصدار **غير المُتصل بالإنترنت** (Offline) من مفكرة جوبيتر، وأسهل طريقة لتثبيته محليًا هي من خلال **أناكوندا** (Anaconda) وهي منصة توزيع مفتوحة المصدر للطلبة والهواة، ويمكنكَ تنزيلها وتثبيتها من الرابط التالي:

https://www.anaconda.com/products/distribution

وسيتم تثبيت لغة البايثون ومفكرة جوبيتر تلقائيًا.

**لفتح مفكرة جوبيتر (Jupyter Notebook):**

- < اضغط على Start (بدء)، 1 ثم اضغط على Anaconda3 (أناكوندا 3). 2
- < اختر Jupyter Notebook (مفكرة جوبيتر). 3
- < ستظهر الصفحة الرئيسة لمفكرة جوبيتر في المتصفح.

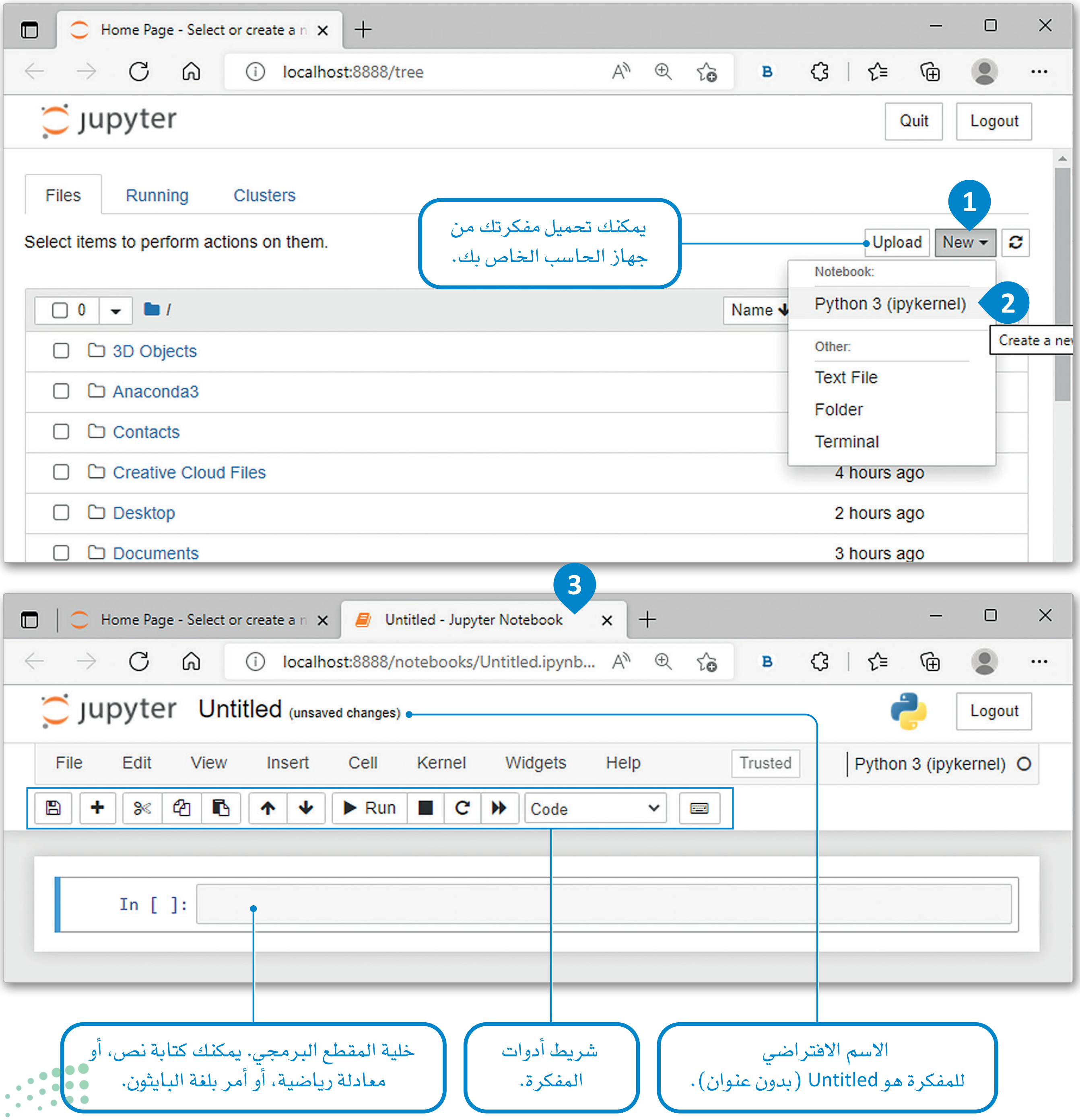



شكل 1.16: الصفحة الرئيسة لمفكرة جوبيتر


وزارة التعليم
Ministry of Education
2025 - 1447

## لإنشاء مفكرة جوبيتر جديدة:

- في الزاوية اليمنى العلوية من شاشتك، اضغط على New (جديد). 1
- حدِّد Python 3 (ipykernel) (بايثون 3). 2
- سيتم فتح المفكرة الخاصة بك في علامة تبويب جديدة في المتصفح الخاص بك. 3




شكل 1.17: إنشاء مفكرة جوبيتر جديدة

وزارة التعليم
Ministry of Education
2025 - 1447

الآن وبعد أن أصبحت مُفكرتك جاهزة، حان الوقت لكتابة برنامجك الأول وتشغيله فيها.

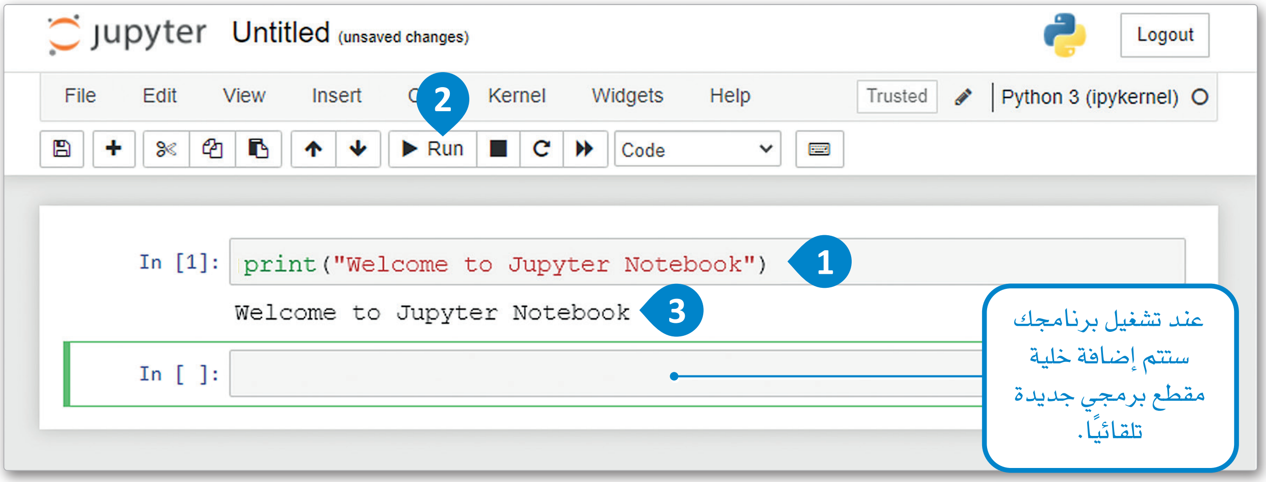
**يمكنك الحصول على العديد من الخلايا المختلفة التي تحتاجها في نفس المفكرة حيث تحتوي كل خلية على مقطعها البرمجي الخاص.**

### لإنشاء برنامج في مفكرة جوبيتر:

< أكتب الأوامر داخل خلية المقطع البرمجي. 1

< اضغط على Run (تشغيل). 2

< سيتم عرض النتيجة تحت الأوامر. 3




شكل 1.18: إنشاء برنامج في مفكرة جوبيتر

حان الوقت لحفظ مفكّرتك.

**يُمكنك تشغيل برنامجك بالضغط على Shift + Enter ↵.**

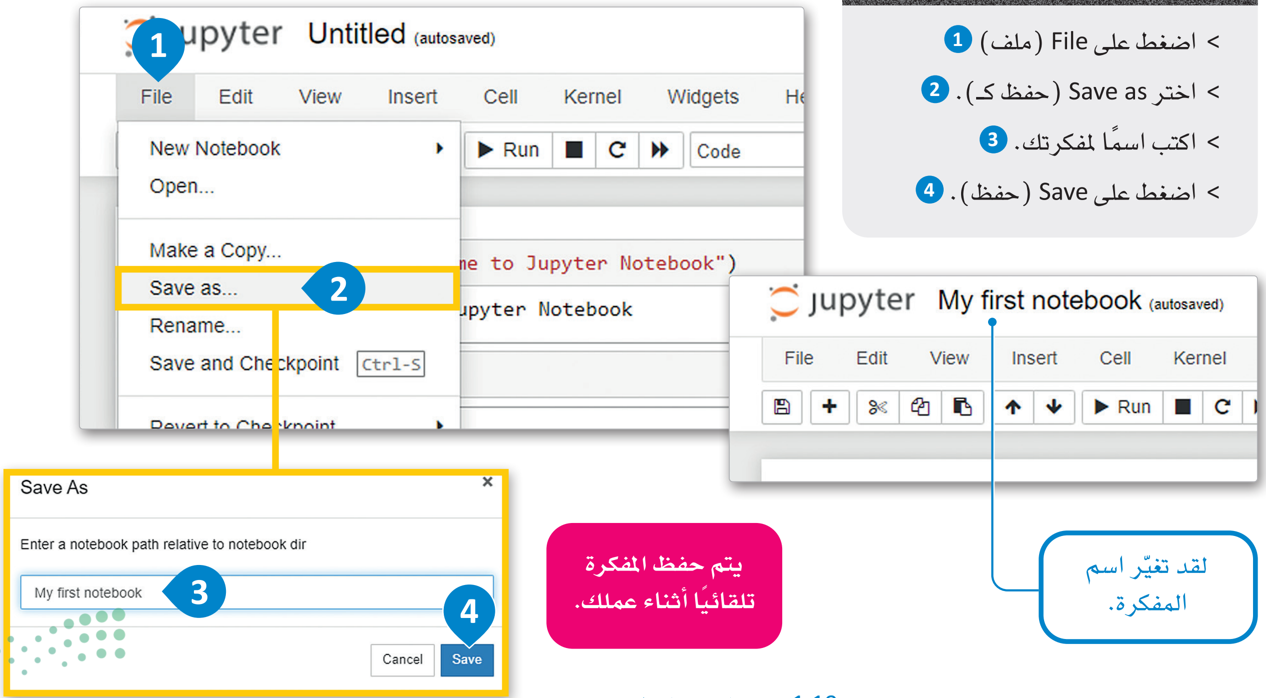
### لحفظ المفكرة الخاصة بك:

< اضغط على File (ملف) 1

< اختر Save as (حفظ كـ). 2

< اكتب اسمًا لمفكرتك. 3

< اضغط على Save (حفظ). 4




شكل 1.19: حفظ المفكرة الخاصة بك

لتشاهد المثال في الشكل 1.15 في مفكرة جوبيتر:

1. أنشئ المُكدّس لتخزين مجموعة من الأرقام (1، 21، 32، 45).
2. استخدِم عملية **حذف عنصر** (Pop) من المُكدّس مرتين لحذف العنصرين الأخيرين منه.
3. استخدِم عملية **إضافة عنصر** (Push) إلى المُكدّس لإضافة عنصر جديد إليه.

```
myStack=[1,21,32,45]
print("Initial stack: ", myStack)
print(myStack.pop())
print(myStack.pop())
print("The new stack after pop: ", myStack)
myStack.append(78)
print("The new stack after push: ", myStack)
```

تُستخدَم الدالة ( print(myStack.pop() لعرض القيم المُسترجَعة من دالة ( )myStack.Pop.

```
Initial stack: [1, 21, 32, 45]
45
32
The new stack after pop: [1, 21]
The new stack after push: [1, 21, 78]
```

```
myStack=[1,21,32,45]
print("Initial stack:", myStack)
a=len(myStack)
print("size of stack",a)
# empty the stack
for i in range(a):
   myStack.pop()
print(myStack)
myStack.pop()
```

تُستخدَم الدالة len لعرض طول المُكدّس.

يُستخدَم هذا الأمر لحذف كل العناصر من المُكدّس.

```
Initial stack: [1, 21, 32, 45]
size of stack 4
[]
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Input In [3], in <cell line: 9>()
      7     myStack.pop()
      8 print(myStack)
----> 9 myStack.pop()

IndexError: pop from empty list
```

يظهر الخطأ؛ لأن المُكدّس فارغ وأنت كتبت أمر حذف عنصر منه.

**خطأ الفهرس IndexError**

ستلاحظ ظهور خطأ عندما كتبتَ أمر حذف عنصر من المُكدّس الفارغ وتسبب هذا في **غَيْض المُكدّس** (Stack Underflow). عليك دومًا التحقق من وجود عناصر في المُكدّس قبل محاولة حذف عنصر منه.

وزارة التعليم
Ministry of Education
2025 - 1447

في البرنامج التالي ستنشئ مُكدّسًا جديدًا وتضيف العناصر إليه، أو تحذفها منه، سيظهر بالبرنامج قائمة تطلب منك تحديد الإجراء الذي تود القيام به في كل مرة.

- لإضافة عنصر إلى المُكدّس، اضغط على الرقم 1 من قائمة البرنامج.
- لحذف عنصر من المُكدّس، اضغط على الرقم 2 من قائمة البرنامج.
- للخروج من البرنامج، اضغط على الرقم 3 من قائمة البرنامج.

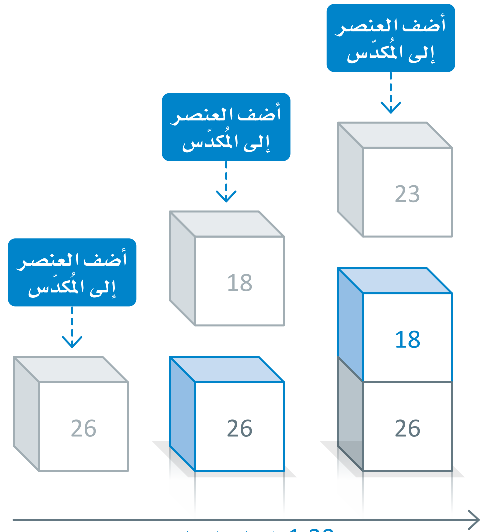
```
def push(stack,element):
    stack.append(element)
def pop(stack):
    return stack.pop()
def isEmpty(stack):
    return len(stack)==0
def createStack():
    return []

newStack=createStack()
while True:
    print("The stack so far is:",newStack)
    print("-----------------------------")
    print("Choose 1 for push")
    print("Choose 2 for pop")
    print("Choose 3 for end")
    print("-----------------------------")
    choice=int(input("Enter your choice: "))
    while choice!=1 and choice!=2 and choice!=3:
        print ("Error")
        choice=int(input("Enter your choice: "))
    if choice==1:
        x=int(input("Enter element for push: "))
        push(newStack,x)
    elif choice==2:
        if not isEmpty(newStack):
            print("The pop element is:",pop(newStack))
        else:
            print("The stack is empty")
    else:
        print("End of program")
        break;
```


وزارة التعليم
Ministry of Education
2025 - 1447

نَفِّذ البرنامج السابق كما يلي:

- أنشئ مُكدّسًا من ثلاثة أرقام.
- أضف العناصر إلى المُكدّس.

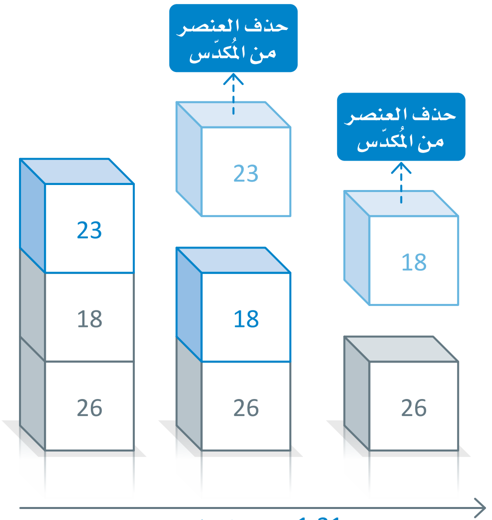
```
The stack so far is: []
-----------------------------
Choose 1 for push
Choose 2 for pop
Choose 3 for end
-----------------------------
Enter your choice: 1
Enter element for push: 26
The stack so far is: [26]
-----------------------------
Choose 1 for push
Choose 2 for pop
Choose 3 for end
-----------------------------
Enter your choice: 1
Enter element for push: 18
The stack so far is: [26, 18]
-----------------------------
Choose 1 for push
Choose 2 for pop
Choose 3 for end
-----------------------------
Enter your choice: 1
Enter element for push: 23
The stack so far is: [26, 18, 23]
-----------------------------
```




شكل 1.20: إضافة العناصر

يمكنك الآن حذف عنصرين من المُكدّس، ثم الخروج من البرنامج.

```
Choose 1 for push
Choose 2 for pop
Choose 3 for end
-----------------------------
Enter your choice: 2
The pop element is: 23
The stack so far is: [26, 18]
-----------------------------
Choose 1 for push
Choose 2 for pop
Choose 3 for end
-----------------------------
Enter your choice: 2
The pop element is: 18
The stack so far is: [26]
-----------------------------
Choose 1 for push
Choose 2 for pop
Choose 3 for end
-----------------------------
Enter your choice: 3
End of program
```




شكل 1.21: حذف العناصر

وزارة التعليم
Ministry of Education
2025 - 1447

## الطابور Queue

هيكل البيانات التالي الذي سنستعرضه هو الطابور. تُصادِف عادةً طوابير في حياتك اليومية. الطابور الأكثر شيوعًا هو طابور انتظار السيارات عند إشارة المرور. عندما تتحول إشارة المرور إلى اللون الأخضر، ستكون السيارة التي دخلت إلى الطابور أولًا هي نفسها التي تخرج منه أولًا. الطابور هو هيكل البيانات الذي يَتبع قاعدة **المُضاف أولًا يَخرُج أولًا** (First In First Out - FIFO)، مما يعني أن كل عنصر في الطابور يُقدَّم بالترتيب نفسه الذي وصل به إلى الطابور.

## العمليات في الطابور Operations on the Queue

هناك عمليتان رئيستان في الطابور:

- **إضافة عنصر للطابور** (Enqueue): تُستخدَم العملية لإضافة عنصر في آخر الطابور.
- **حذف عنصر من الطابور** (Dequeue): تُستخدَم العملية لحذف عنصر من مقدمة الطابور.

### مؤشرات الطابور Queue Pointers

يحتوي الطابور على مؤشرين:

- **المؤشر الأمامي** (Front Pointer): يُشير إلى العنصر الأول في الطابور.
- **المؤشر الأخير** (Rear Pointer): يُشير إلى العنصر الأخير في الطابور.

**قاعدة المُضاف أولًا يَخرُج أولًا (First In First Out (FIFO) Rule):**

العنصر الأول المُضاف إلى القائمة يُعالَج أولًا، والعنصر الأحدث يُعالَج آخرًا.

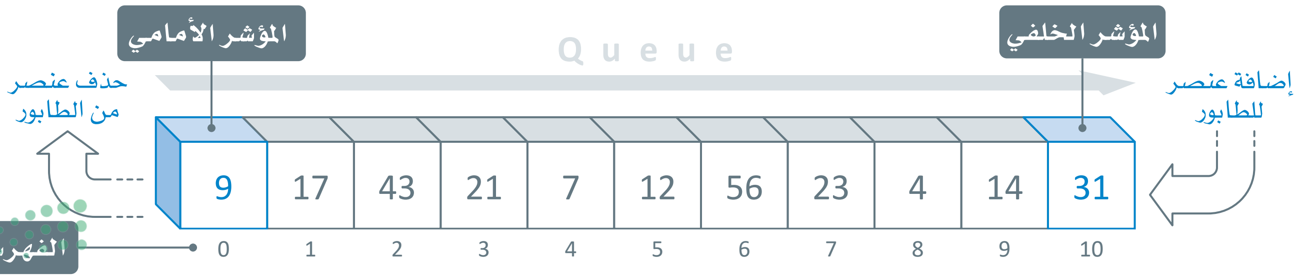
الفرق بين المُكدّس والطابور هو أنه في المُكدس تتم إضافة وحذف العنصر من نفس الجانب، وفي الطابور تتم الإضافة من جانب، بينما يتم الحذف من الجانب الآخر. وهكذا، عند الحذف في المُكدّس، يُحذف العنصر المُضاف آخرًا، بينما في الطابور، يُحذف العنصر المُضاف أولًا.

**المؤشر (Pointer):**

المؤشر هو مُتغير يُخزّن أو يُشير إلى عنوان مُتغير آخر. المؤشر يشبه رقم الصفحة في فهرس الكتاب الذي يُسهِّل على القارئ الوصول إلى المحتوى المطلوب.

**الفهرس (Index):**

الفهرس هو رقم يُحدِّد موضع العنصر في هيكل البيانات.



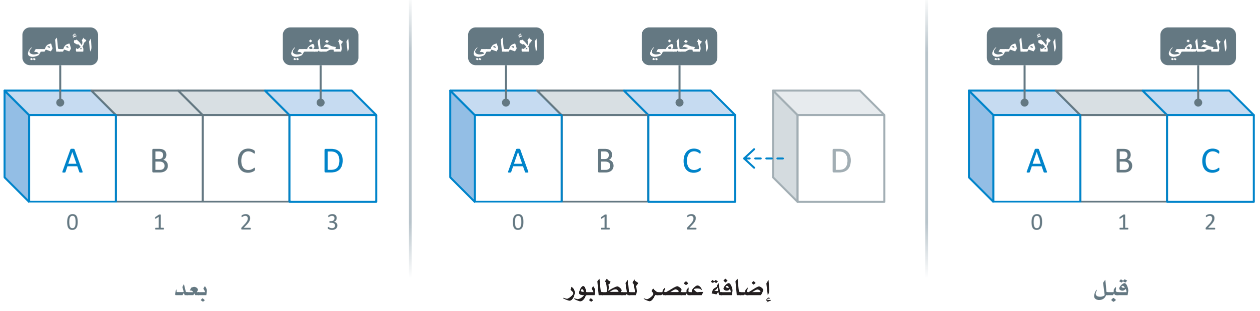

شكل 1.22: العمليات في الطابور

## عملية إضافة عنصر للطابور Enqueue Operation

> لا يمكنك إضافة عنصر أو حذفه من وسط الطابور.

يُطلق على عملية إضافة عنصر جديد إلى الطابور اسم **إضافة عنصر للطابور** (Enqueue). لإضافة عنصر جديد إلى الطابور:

- تتم زيادة قيمة المؤشر الخلفي بقيمة واحد بحيث يشير إلى موضع العنصر الجديد الذي سيُضاف.
- تتمّ إضافة العنصر.




شكل 1.23: عملية إضافة عنصر للطابور

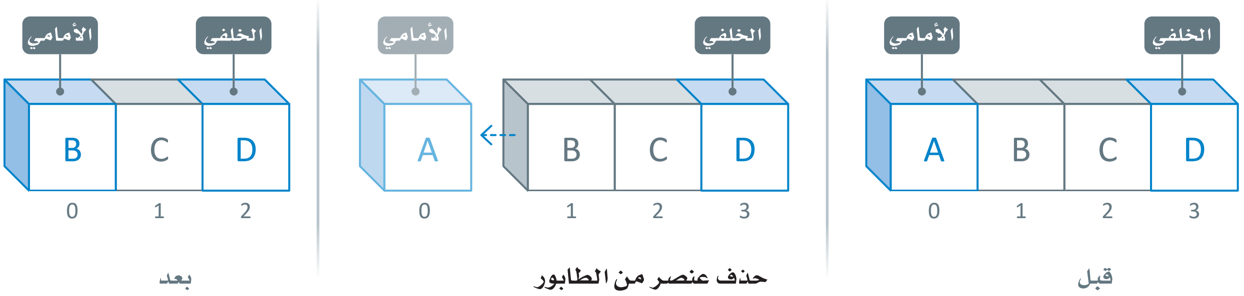
## عملية حذف عنصر من الطابور Dequeue Operation

> قبل أي إجراء عليك التحقق مما إذا كانت هناك مساحة فارغة في الطابور لإضافة عنصر جديد، أو توافر عنصر واحد على الأقل لتصديره.

يُطلق على عملية حذف عنصر من الطابور اسم **حذف عنصر من الطابور** (Dequeue).

لحذف عنصر من الطابور:

- يُحذف العنصر المُشار إليه بالمؤشر الأمامي.
- تتم زيادة قيمة المؤشر الأمامي بقيمة واحد بحيث يشير إلى العنصر الجديد التالي في الطابور.




شكل 1.24: عملية حذف عنصر من الطابور

## الطابور في لغة البايثون Queue in Python

يمكن تمثيل الطابور بعدة طرائق متنوعة في لغة البايثون منها **القوائم** (Lists). ويرجع ذلك إلى حقيقة أن القائمة تمثل مجموعة من العناصر الخطيّة، كما يمكن إضافة عنصر في نهاية القائمة وحذف عنصر من بداية القائمة.

ستتعلّم فيما يلي الصيغ العامة لبعض العمليات التي يمكن تنفيذها على الطابور:

**جدول 1.3: طرائق الطابور**

| الطريقة | الوصف |
|---|---|
| listName.append(x) | تضيف العنصر x إلى القائمة التي تمثل الطابور. |
| listName.pop(0) | تحذف العنصر الأول من القائمة. |

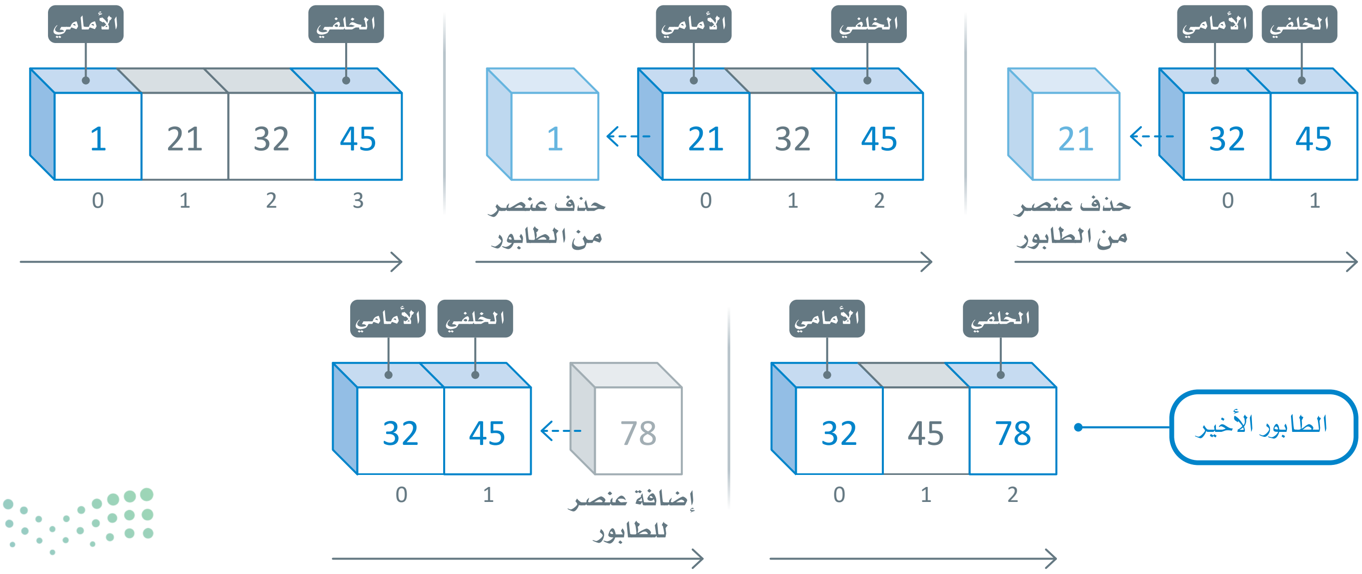
تُستخدَم طريقة ()listName.pop لكلٍ من هياكل بيانات المُكدّس والطابور. عندما تُستخدَم مع المُكدّس، لا تتطلب الطريقة أي مُعامِل. بينما تتطلب الطريقة إضافة صفر إلى المُعامل عندما تُستخدم مع الطابور: listName.pop(0). الفرق بين الدالتين مُوضّح في الجدول 1.4 أدناه.

**جدول 1.4: طريقة ()listName.pop مقابل طريقة listName.pop(0)**

| الطريقة | الوصف |
|---|---|
| listName.pop() | إذا كان مُعامِل الدالة فارغًا، يُحذف العنصر الأخير من نهاية القائمة التي تمثل المُكدّس. |
| listName.pop(0) | إذا كان مُعامِل الدالة صفرًا، يُحذف العنصر الأول من القائمة التي تمثل الطابور. |

سنستعرض لك مثالًا على تطبيق الطابور في لغة البايثون:

- أنشئ طابورًا لتخزين مجموعة من الأرقام (1، 21، 32، 45).
- استخدم عملية حذف عنصر من الطابور مرتين لحذف العنصرين الأوَّلَين منه.
- استخدم عملية إضافة عنصر إلى الطابور لإضافة عنصر جديد إليه.




شكل 1.25: مثال توضيحي لمفهوم الطابور


وزارة التعليم
Ministry of Education
2025 - 1447

لبرمجة الخطوات الموضحة بالأعلى بلغة البايثون، ستَستخدِم قائمة البايثون لتنفيذ هيكل الطابور، كما فعلت في المُكدّس.

```
myQueue=[1,21,32,45]
print("Initial queue: ", myQueue)
myQueue.pop(0)
myQueue.pop(0)
print("The new queue after pop: ", myQueue)
myQueue.append(78)
print("The new queue after push: ", myQueue)
```

```
Initial queue: [1, 21, 32, 45]
The new queue after pop: [32, 45]
The new queue after push: [32, 45, 78]
```

لكي تشاهد ما قد يحدث عندما تحاول حذف عنصر من طابور فارغ، عليكَ أولًا أن تُفرِغ الطابور من العناصر.

```
myQueue=[1,21,32,45]
print("Initial queue: ", myQueue)
a=len(myQueue)
print("size of queue ",a)
# empty the queue
for i in range(a):
   myQueue.pop(0)
print(myQueue)
myQueue.pop(0)
```

```
Initial queue: [1, 21, 32, 45]
size of queue 4
[]
---------------------------------------------------------------------
IndexError                          Traceback (most recent call last)
Input In [6], in <cell line: 9>()
      7     myQueue.pop()
      8 print(myQueue)
----> 9 myQueue.pop()

IndexError: pop from empty list
```

ظهر الخطأ؛ لأنك حاولت حذف عنصر من طابور فارغ.

عليك أن تتحقق دومًا من وجود عناصر في الطابور قبل محاولة حذف عنصر منه.

## تطبيقات على الطابور Queue Applications

أحد الأمثلة على تطبيقات الطابور في علوم الحاسب هو طابور الطباعة. على سبيل المثال، لديك معمل حاسب به 30 جهاز حاسب متصلين بطابعة واحدة. عندما يرغب الطلبة في طباعة المُستندات، ستشكّل مهامّ الطّباعةِ طابورًا لمعالجتها وفق قاعدة **المُضاف أولًا يَخرُج أولًا** (FIFO)، أي أنّ تلك المهام ستُنجَزُ بالترتيب الزمني الذي أُرسلَت به إلى الطابعة. المهمة المُرسلة أولًا ستُطبع قبل المهمة المُرسلة بعدها ولن تُطبع المهمة في نهاية الطابور قبل طباعة كل المهام التي قبلها. عندما تنتهي الطابعة من أحد الأوامر، ستبحث في الطابور لمعرفة ما إن كانت هناك أوامر أخرى لمعالجتها.

## المُكدّس والطابور باستخدام وحدة الطابور النمطية
## Stack and Queue Using Queue Module

يمكن اعتبار القائمة في لغة البايثون بمثابة طابور وكذلك مُكدّس. تُقدِّم لغة البايثون **الوحدة النمطية للطابور** (Queue Module) وهي طريقة أخرى لتنفيذ هيكليّ البيانات الموضحين. تتضمن الوحدة النمطية للطابور بعض الدوال الجاهزة للاستخدام التي يمكن تطبيقها على كل من المُكدّس والطابور.

**جدول 1.5: طرائق الوحدة النمطية للطابور**

| الطريقة | الوصف |
|---|---|
| queueName=queue.Queue( ) | تنشئ طابورًا جديدًا اسمه queueName. |
| queueName.put(x) | تضيف العنصر x إلى الطابور. |
| queueName.qsize( ) | تعود بقيمة حجم الطابور. |
| queueName.get( ) | تعرض وتحذف العنصر الأول من الطابور والعنصر الأخير من المُكدّس. |
| queueName.full( ) | تعود بقيمة True (صحيح) إن كان الطابور ممتلئًا، وقيمة False (خطأ) إن كان الطابور فارغًا، ويمكن تطبيقها على المُكدّس كذلك. |
| queueName.empty( ) | تعود بقيمة True (صحيح) إن كان الطابور فارغًا والقيمة False (خطأ) إن كان الطابور ممتلئًا، ويمكن تطبيقها على المُكدّس كذلك. |

**تُستخدَم طرائق الوحدة النمطية للطابور مع كلٍ من المُكدّس والطابور.**

ستَستخدِم وحدة الطابور النمطية لإنشاء طابور.

في هذا المثال عليك:

- استيراد الوحدة النمطية **للطابور** (Queue) لاستخدام طرائق الطابور.
- إنشاء طابور فارغ باسم myQueue (طابوري).
- إضافة العناصر a، b، c، d، e إلى الطابور myQueue (طابوري).
- طباعة عناصر الطابور.

```
from queue import *

myQueue = Queue()
# add the elements in the queue
myQueue.put("a")
myQueue.put("b")
myQueue.put("c")
myQueue.put("d")
myQueue.put("e")

# print the elements of the queue
for element in list(myQueue.queue):
   print(element)
```

```
a
b
c
d
e
```

عليك استيراد الوحدة النمطية للطابور في بداية المقطع البرمجي.


وزارة التعليم
Ministry of Education
2025 - 1447

أنشئ طابورًا مُكوَّنًا من خمس قيم يقوم المُستخدِم بإدخالها أثناء تنفيذ البرنامج، ثم اطبع هذه القيم، وفي النهاية اطبع حجم الطابور.

```
from queue import *

myQueue = Queue()

# the user enters the elements of the queue for i in range(5):
for i in range(5):
   element=input("enter queue element: ")
   myQueue.put(element)

# print the elements of the queue
for element in list(myQueue.queue):
   print(element)

print ("Queue size is: ",myQueue.qsize())
```

```
enter queue element: 5
enter queue element: f
enter queue element: 12
enter queue element: b
enter queue element: 23
5
f
12
b
23
Queue size is: 5
```

أنشئ برنامجًا للتحقق مما إذا كان الطابور فارغًا أم ممتلئًا.

```
from queue import *

myQueue = Queue()

myQueue.put("a")
myQueue.put("b")
myQueue.put("c")
myQueue.put("d")
myQueue.put("e")

checkFull=myQueue.full()
print("Is the queue full? ", checkFull)
checkEmpty= myQueue.empty()
print("Is the queue empty? ", checkEmpty)
```

```
Is the queue full? False
Is the queue empty? False
```

كما ذُكِر من قبل فإن الوحدة النمطية للطابور تحتوي على بعض الوظائف الجاهزة للاستخدام مع المُكدّس أو الطابور. الجدول 1.6 يوضِّح وظائف الوحدة التي يُمكن استخدامها مع هيكل بيانات المُكدّس.

**جدول 1.6: طرائق الوحدة المُستخدمة للمُكدّس**

| الطريقة | الوصف |
|---|---|
| stackName=queue.LifoQueue() | تنشئ مُكدّسًا جديدًا اسمه stackName. |
| stackName.get() | تحذف العنصر الأخير من المُكدّس. |

ستَستخدِم وحدة الطابور لإنشاء مُكدّس فارغ.

```
from queue import *

myStack = LifoQueue()

myStack.put("a")
myStack.put("b")
myStack.put("c")
myStack.put("d")
myStack.put("e")

for i in range(5):
   k=myStack.get()
   print(k)

# empty the stack
checkEmpty= myStack.empty()
print("Is the stack empty?", checkEmpty)
```

تذكّر أن العمليات في المُكدّس تعمل وفقًا لقاعدة المُضاف آخرًا يَخرُج أولًا (LIFO).

عند استخدام دالة get مع الطابور، ستَستنِد عمليات الاستدعاء والطباعة إلى قاعدة المُضاف أولًا يَخرُج أولًا (FIFO).

```
e
d
c
b
a
Is the stack empty? True
```

## مثال: الطباعة Print

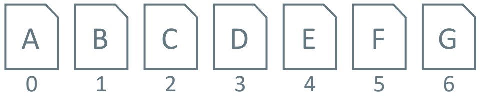
يظهر أمامك في المثال التالي محاكاة لطابور الطباعة في الطابعة. عندما يُرسِل المُستخدِمون أوامر طباعة، تُضاف إلى طابور الطباعة. تَستخدِم الطابعة هذا الطابور لتحديد الملف الذي سيُطبع أولًا.

- افترض أن سعة الطابعة هي فقط 7 ملفات، ولكن في الوقت نفسه، تحتاج إلى طباعة 10 ملفات من الملف A إلى الملف J.
- اكتب برنامجًا يُمثّل طابور الطباعة منذ بدء أمر الطباعة الأول A حتى الانتهاء من كل أوامر الطباعة.
- أضف اللبنة التي تؤكد أن طابور أوامر الطباعة فارغ.

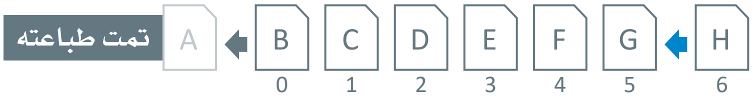
يُمكنك استخدام الخوارزمية الآتية:

1 أنشئْ طابور أوامر الطباعة.

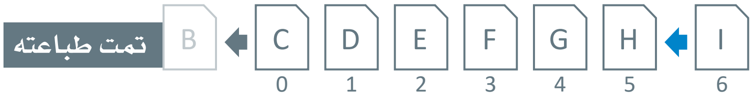
2 أدرِج الملفات من A إلى G في طابور أوامر الطباعة.



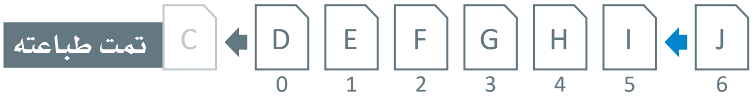

3 أخرِج الملف A وأدرج الملف H.




4 أخرِج الملف B وأدرج الملف I.




5 أخرِج الملف C وأدرج الملف J.




6 أخرِج الملفات التي تمت طباعتها واحدًا تلو الآخر (D-E-F-G-H-I-J).

```python
# import the queue library
from queue import *
# import the time library to use the sleep function
import time
# initialize the variables and the queue
printDocument = " "
printQueueSize = 0
printQueueMaxSize = 7
printQueue = Queue(printQueueMaxSize)
# add a document to print the queue
def addDocument(document):
    printQueueSize = printQueue.qsize()
    if printQueueSize == printQueueMaxSize:
        print("!! ", document, " was not sent to print queue.")
        print("The print queue is full.")
        print()
        return
    printQueue.put(document)
    time.sleep(0.5) #Wait 5.0 seconds
    print(document, " sent to print queue.")
    printQueueSizeMessage()
# print a document from the print queue
def printDocument():
    printQueueSize = printQueue.qsize()
    if printQueueSize == 0:
        print("!! The print queue is empty.")
```

```
        print()
        return
    printDocument = printQueue.get()
    time.sleep(1) # wait one second
    print ("OK - ", printDocument, " is printed.")
    printQueueSizeMessage()
# print a message with the size of the queue
def printQueueSizeMessage():
    printQueueSize = printQueue.qsize()
    if printQueueSize == 0:
        print ("There are no documents waiting for printing.")
    elif printQueueSize == 1:
        print ("There is 1 document waiting for printing.")
    else:
        print ("There are ", printQueueSize, " documents waiting for printing.")
    print()
# the main program
# send documents to the print queue for printing
addDocument("Document A")
addDocument("Document B")
addDocument("Document C")
addDocument("Document D")
addDocument("Document E")
addDocument("Document F")
addDocument("Document G")
printDocument()
addDocument("Document H")
printDocument()
addDocument("Document I")
printDocument()
addDocument("Document J")
addDocument("Document K")
printDocument()
printDocument()
printDocument()
printDocument()
printDocument()
printDocument()
printDocument()
printDocument()
```

```
Document A sent to print queue.
There is 1 document waiting for printing.

Document B sent to print queue.
There are 2 documents waiting for printing.

Document C sent to print queue.
There are 3 documents waiting for printing.
```

```
Document D sent to print queue.
There are 4 documents waiting for printing.

Document E sent to print queue.
There are 5 documents waiting for printing.

Document F sent to print queue.
There are 6 documents waiting for printing.

Document G sent to print queue.
There are 7 documents waiting for printing.

OK - Document A is printed.
There are 6 documents waiting for printing.

Document H sent to print queue.
There are 7 documents waiting for printing.

OK - Document B is printed.
There are 6 documents waiting for printing.

Document I sent to print queue.
There are 7 documents waiting for printing.

OK - Document C is printed.
There are 6 documents waiting for printing.

Document J sent to print queue.
There are 7 documents waiting for printing.

!! Document K was not sent to print queue.
The print queue is full.

OK - Document D is printed.
There are 6 documents waiting for printing.

OK - Document E is printed.
There are 5 documents waiting for printing.

OK - Document F is printed.
There are 4 documents waiting for printing.

OK - Document G is printed.
There are 3 documents waiting for printing.

OK - Document H is printed.
There are 2 documents waiting for printing.

OK - Document I is printed.
There is 1 document waiting for printing.

OK - Document J is printed.
There are no documents waiting for printing.

!! The print queue is empty.
```

## هياكل البيانات الثابتة والمتغيرة Static and Dynamic Data Structures

سبق توضيح أن هياكل البيانات هي طريقة فعالة لتخزين البيانات وتنظيمها، وبالإضافة إلى ما تعلمته حول تصنيف هياكل البيانات إلى أوليّة وغير أوليّة، فإنه يمكن تصنيفها أيضًا إلى **ثابتة** (Static) و**متغيرة** (Dynamic).

### هياكل البيانات الثابتة Static Data Structure

في البيانات الثابتة، يكون حجم الهيكل ثابتًا، وتُخزَّن عناصر البيانات في مواقع الذاكرة المتجاورة. تُعدُّ **المصفوفة** (Array) المثال الأبرز لهياكل البيانات الثابتة.

### هياكل البيانات المتغيرة Dynamic Data Structure

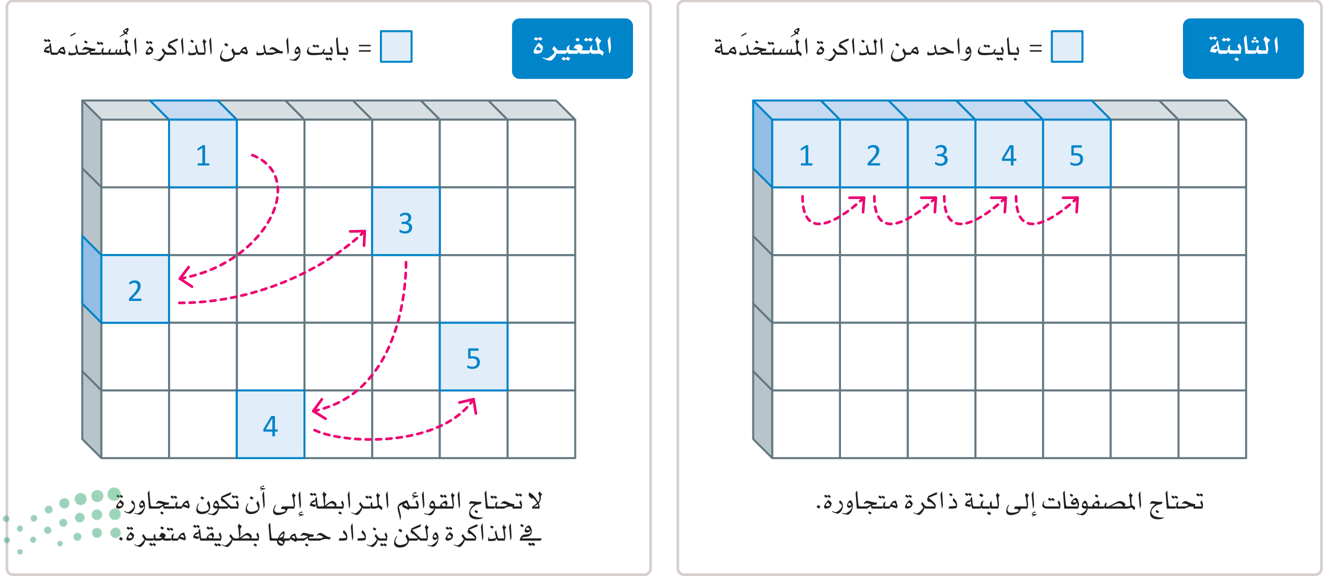
في هياكل البيانات المتغيرة، لا يكون حجم الهيكل ثابتًا ولكن يمكن تعديله أثناء تنفيذ البرنامج، حسب العمليات المُنفَّذة عليه. تُصمَّم هياكل البيانات المتغيرة لتسهيل تغيير حجم هياكل البيانات أثناء التشغيل. وتُعدُّ **القائمة المترابطة** (Linked List) المثال الأبرز لهياكل البيانات المتغيرة.

**جدول 1.7: مقارنة بين هياكل البيانات الثابتة والمتغيرة**

| | الثابتة | المتغيرة |
|---|---|---|
| حجم الذاكرة | حجم الذاكرة ثابت. | يمكن تغيير حجم الذاكرة أثناء التشغيل. |
| أنواع ذاكرة التخزين | تُخزَّن العناصر في مواقع متجاورة في الذاكرة الرئيسة. | تُخزَّن العناصر في مواقع عشوائية في الذاكرة الرئيسة. |
| سرعة الوصول إلى البيانات | أسرع. | أبطأ. |

## تخصيص الذاكرة Memory Allocation

تنتمي **القوائم المترابطة** (Linked Lists) إلى هياكل البيانات المتغيرة، وهذا يعني أن عُقد القائمة المترابطة لا تُخزَّن في مواقع الذاكرة المتجاورة مثل البيانات في المصفوفات. ولهذا السبب، تحتاج إلى تخزين المؤشر من عُقدة إلى أخرى.



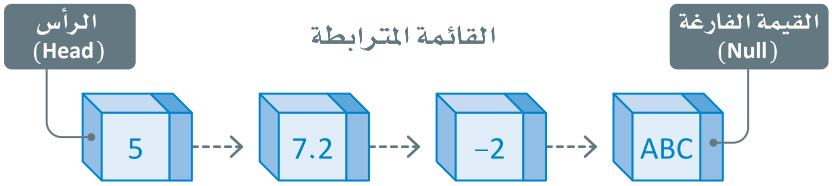

شكل 1.26: مثال على تخصيصيّ الذاكرة الثابتة والمتغيرة


وزارة التعليم
Ministry of Education
2025 - 1447

## القائمة المترابطة Linked List

القائمة المترابطة هي نوع من هياكل البيانات الخطيَّة، وهي واحدة من هياكل البيانات الأكثر شهرة في البرمجة. القائمة المترابطة تشبه سلسلة من العُقد. تحتوي كل عُقدة على حقلين: حقل البيانات حيث تُخزن البيانات، وحقل يحتوي على المؤشر الذي يُشير إلى العُقدة التالية. يُستثنى من هذا العُقدة الأخيرة التي لا يحمل فيها حقل العنوان أي بيانات. إحدى مزايا القائمة المترابطة هي أن حجمها يزداد أو يقل بإضافة أو حذف العُقد.

**القائمة المترابطة (Linked List):**
القائمة المترابطة هي نوع من هياكل البيانات الخطيَّة التي تشبه سلسلة من العُقد.



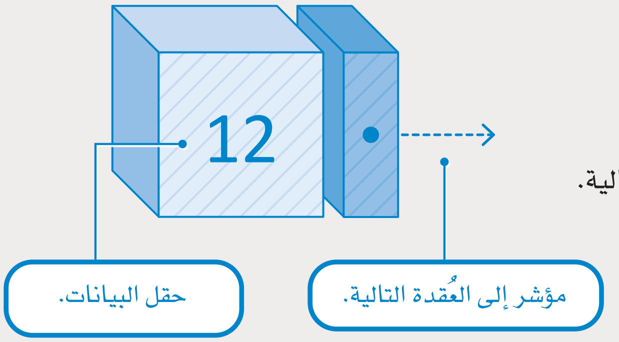

شكل 1.27: رسم توضيحي للقائمة المترابطة

**العُقدة (Node):**
العُقدة هي اللبنة الفردية المُكوِّنة لهيكل البيانات وتحتوي على البيانات ورابط واحد أو أكثر من الروابط التي تربطها بالعُقد الأخرى.

### العُقدة Node

تتكون كل عُقدة في القائمة المترابطة من جزئين:

- الجزء الأول يحتوي على البيانات.
- الجزء الثاني يحتوي على مؤشر يُشير إلى العُقدة التالية.

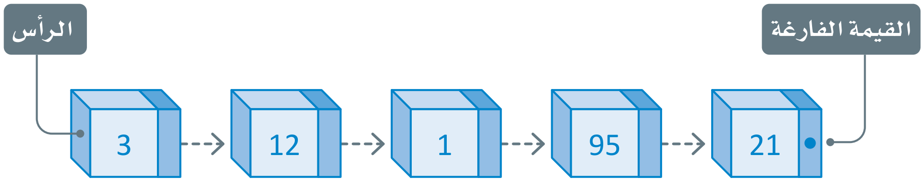



شكل 1.28: رسم توضيحي للعُقد

لقراءة محتوى عُقدة محددة، عليك المرور على كل العُقد السابقة.

لتشاهد مثالًا على القائمة المترابطة للأعداد الصحيحة.

تتكون القائمة المترابطة من خمس عُقد.




شكل 1.29: رسم توضيحي يُمثِّل قائمة مترابطة للأعداد الصحيحة

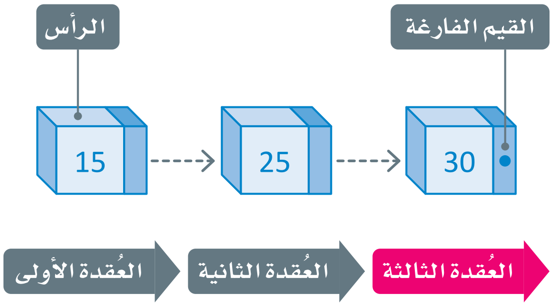
القيمة الفارغة تعني أنها بلا قيمة، أو غير مُحدَّدة، أو فارغة. على الرغم من أنه في بعض الأحيان يُستخدم الرقم 0 للإشارة إلى القيمة الفارغة، إلا أنه رقم محدَّد وقد يشير إلى قيمة حقيقية.

العُقد في القائمة لا يكون لها اسم، وما تعرفه عنها هو عنوانها (الموقع الذي تخزن فيه العُقدة في الذاكرة). للوصول إلى أي عُقدة بالقائمة، تحتاج فقط إلى معرفة عنوان العُقدة الأولى. ثم تتبع سلسلة العُقد للوصول إلى العُقدة المطلوبة.

وزارة التعليم
Ministry of Education
2025 - 1447

على سبيل المثال، إن كنت ترغب في الوصول إلى العُقدة الثالثة في القائمة لمعالجة البيانات التي تحتوي عليها، عليك البدء من العُقدة الأولى في القائمة، ومن العُقدة الأولى للوصول إلى الثانية، ومن الثانية للوصول إلى الثالثة.

- عنوان العُقدة الأولى مُخزَّن في مُتغير خاص (مُستقِل) يُطلق عليه عادةً **الرأس** (Head).
- قيمة مؤشر العُقدة الأخيرة في القائمة **قيمة فارغة** (Null)، ويُمثَّل بالرمز ●.
- عندما تكون القائمة فارغة، يشير مؤشر الرأس إلى **القيمة الفارغة** (Null).



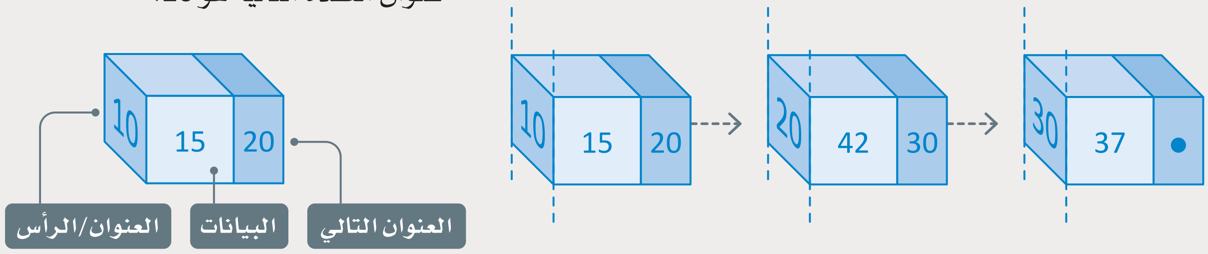
شكل 1.30: الوصول إلى العُقدة الثالثة في القائمة المترابطة

إليك مثالًا توضيحيًا على القائمة المترابطة في الشكل 1.31، كما ذُكر من قبل فإن كل عُقدة تتكون من بيانات ومؤشر يشير إلى العُقدة التالية، بحيث تُخزّن كل عُقدة في الذاكرة في عنوان مُحدَّد.

لتربط العُقدة السابقة بالعُقدة التالية بقيمة بيانات 42، التي بدورها تُشير إلى العُقدة الثالثة والأخيرة عند عنوان 30 بقيمة بيانات 37.

مثال على العُقدة:

- بيانات العُقدة هي الرقم 15.
- عنوان العُقدة في الذاكرة هو 10.
- عنوان العُقدة التالية هو 20.



شكل 1.31: المؤشرات في القائمة المترابطة

## جدول 1.8: الاختلافات بين القائمة والقائمة المترابطة

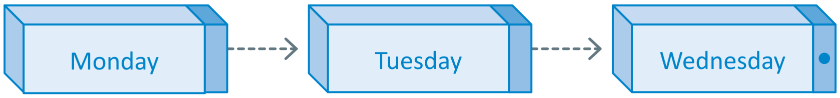
| الاختلافات | القائمة | القائمة المترابطة |
| --- | --- | --- |
| طريقة تخزين الذاكرة | المواقع متجاورة في الذاكرة. | المواقع عشوائية في الذاكرة. |
| الهيكل | يمكن الوصول إلى كل عنصر برقم **الفهرس** (Index). | يمكن الوصول إلى العناصر من خلال **المؤشر** (Pointer). |
| الحجم | يُخزَّن كل عنصر تلو الآخر. | تُخزَّن العناصر في صورة عُقد تحتوي على البيانات وعنوان العنصر التالي. |
| استخدام الذاكرة | تُخزَّن البيانات وحدَها في الذاكرة. | تُخزَّن البيانات والمؤشرات في الذاكرة. |
| نوع الوصول إلى البيانات | الوصول العشوائي إلى أي عنصر بالقائمة. | الوصول المتسلسل إلى العناصر. |
| سرعة الإضافة والحذف | بطء إضافة العناصر وحذفها. | سرعة إضافة العناصر وحذفها. |

## القائمة المترابطة في لغة البايثون Linked List in Python

لا تُوفر لغة البايثون نوع بيانات مُحدَّد مُسبقًا للقوائم المترابطة. عليك إنشاء نوع البيانات الخاص بك، أو استخدام مكتبات البايثون التي توفر تمثيلًا لهذا النوع من البيانات. لإنشاء قائمة مترابطة، استخدم فئات البايثون. في المثال الموضح بالشكل 1.32، ستُنشئ قائمة مترابطة مكونة من ثلاث عُقد، كل واحدة تضم يومًا من أيام الأسبوع.

**الفئة (Class):**

الفئة هي هيكل بيانات معرّف بواسطة المُستخدِم، ويحتوي على أعضاء البيانات (السمات Properties)، والطرائق (السلوك Behavior) الخاصة بها. وتُستخدَم الفئات كقوالب لإنشاء الكائنات.




شكل 1.32: مثال على القائمة المترابطة

ستُنشئ أولًا عُقدة باستخدام الفئة.

```
# single node
class Node:
    def __init__(self, data, next=None):
        self.data = data # node data
        self.next = next # Pointer to the next node

# Create a single node
first = Node("Monday")
print(first.data)
```

```
Monday
```

الخطوة التالية هي إنشاء قائمة مترابطة تحتوي على عُقدة واحدة، وهذه المرة ستَستخدِم مؤشر الرأس للإشارة إلى العُقدة الأولى.

```
# single node
class Node:
  def __init__(self, data = None, next=None):
    self.data = data
    self.next = next

# linked list with one head node
class LinkedList:
  def __init__(self):
    self.head = None

# list linked with a single node
Linkedlist1 = LinkedList()
Linkedlist1.head = Node("Monday")
print(Linkedlist1.head.data)
```

```
Monday
```

وزارة التعليم
Ministry of Education
2025 - 1447

أضِفْ الآن المزيد من العُقد إلى القائمة المترابطة.

```
# single node
class Node:
  def __init__(self, data = None, next=None):
    self.data = data
    self.next = next

# an empty linked list with a head node.
class LinkedList:
  def __init__(self):
    self.head = None

# the main program
linked_list = LinkedList()
# the first node
linked_list.head = Node("Monday")
# the second node
linked_list.head.next = Node("Tuesday")
# the third node
linked_list.head.next.next = Node("Wednesday")

# print the linked list
node = linked_list.head
while node:
    print (node.data)
    node = node.next
```

تُستخدَم عبارة while للتنقل من عُقدة إلى أخرى.

```
Monday
Tuesday
Wednesday
```

## إضافة العُقدة إلى القائمة المترابطة

## Add a Node to a Linked List

لتتمكن من إضافة عُقدة جديدة، اتبع الخطوات التالية:

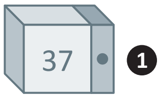
- يجب أن يُشير مؤشر العُقدة الأولى إلى عنوان العُقدة الجديدة، حتى تصبح العُقدة الجديدة هي العُقدة الثانية.
- يجب أن يُشير مؤشر العُقدة الجديدة (الثانية) إلى عنوان العُقدة الثالثة.

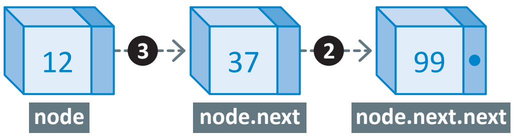
بهذه الطريقة، لن تحتاج إلى تغيير العناصر عند إضافة عنصر جديد في المنتصف. تقتصر العملية على تغيير قيَم العناوين في العُقد التي تُسرِّع من عملية الإضافة في حالة القوائم المترابطة، مقارنةً بحالة القوائم المتسلسلة.

**مثال:**

لديك قائمة مترابطة من عنصرين: 12 و99، وتريد إدراج العنصر 37 كعنصر ثانٍ بالقائمة. في النهاية، سيكون لديك قائمة من ثلاثة عناصر: 12 و37 و99.

1. أنشئ العُقدة الجديدة.




2. اربُط العُقدة 37 بالعُقدة 99.
3. اربُط العُقدة 12 بالعُقدة 37 (تمت إضافة العُقدة الجديدة).

```
# single node
class Node:
  def __init__(self, data = None, next=None):
    self.data = data
    self.next = next

# linked list with one head node
class LinkedList:
  def __init__(self):
    self.head = None

def insertAfter(new, prev):
    # create the new node
    new_node = Node(new)
    # make the next of the new node the same as the next of the previous node
    new_node.next = prev.next
    # make the next of the previous node the new node
    prev.next = new_node

# create the linked list
L_list = LinkedList()

# add the first two nodes
L_list.head = Node(12)
second = Node(99)
L_list.head.next = second

# insert the new node after node 12 (the head of the list)
insertAfter(37, L_list.head)

# print the linked list
node = L_list.head
while node:
    print (node.data)
    node = node.next
```

```
12
37
99
```

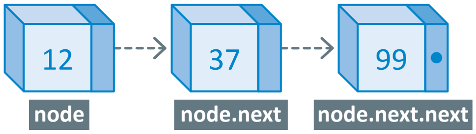
## حذف العُقدة من القائمة المترابطة Delete a Node from a Linked List

لحذف عُقدة، عليك تغيير مُؤشر العُقدة التي تسبق العُقدة المراد حذفها إلى مؤشر العُقدة التي تلي العُقدة المحذوفة. أصبحت العُقدة المحذوفة (الثانية) عبارة عن **بيانات غير مُفيدة** (Useless Data) وستُخصَّص مساحة الذاكرة التي تشغلها لاستخدامات أخرى.

**مثال:**

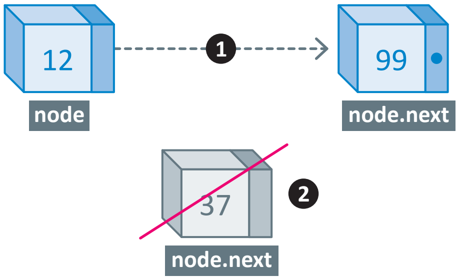
لديك قائمة مترابطة من ثلاثة عناصر: 12 و37 و99، وترغب في حذف العنصر 37. في النهاية، سيكون لديك قائمة من عنصرين: 12 و99.

وزارة التعليم
Ministry of Education
2025 - 1447

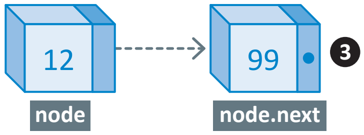



1. اربُط مؤشر العُقدة **12** بالعُقدة **99**.

2. احذف العُقدة **37**.




3. النتيجة النهائية




**إذا كنت تريد حذف العُقدة الأولى من القائمة المترابطة، عليك نقل مؤشر الرأس إلى العُقدة الثانية من القائمة.**

```
# single node
class Node:
  def __init__(self, data = None, next=None):
    self.data = data
    self.next = next

# linked list with one head node
class LinkedList:
  def __init__(self):
    self.head = None

def deleteNode(key, follow):

    # store the head node
    temp = follow.head

    # find the key to be deleted,
    # the trace of the previous node to be changed
    while(temp is not None):
        if temp.data == key:
            break
        prev = temp
        temp = temp.next

    # unlink the node from the linked list
    prev.next = temp.next
    temp = None

# create the linked list
L_list = LinkedList()

# add the first three nodes
L_list.head = Node(12)
second = Node(37)
third = Node(99)
L_list.head.next = second
second.next = third

# delete node 37
deleteNode(37,L_list)

# print the linked list
node = L_list.head
while node:
    print (node.data)
    node = node.next
```

```
12
99
```

# تمرينات

**1**

| حدِّد الجملة الصحيحة والجملة الخاطئة فيما يلي: | صحيحة | خاطئة |
|---|---|---|
| 1. لغة البايثون تُعرِّف هياكل البيانات غير الأوليّة. | ○ | ○ |
| 2. هياكل البيانات الخطيَّة تُخزِّن عناصر البيانات في ترتيب عشوائي فقط. | ○ | ○ |
| 3. إضافة العناصر وحذفها من القائمة المُترابطة (Linked List) أبطأ من القائمة (List). | ○ | ○ |
| 4. يمكن الوصول إلى العناصر في القائمة باستخدام رقم الفهرس فقط. | ○ | ○ |
| 5. يُمكن تغيير حجم هيكل البيانات الثابتة أثناء تنفيذ البرنامج. | ○ | ○ |

**2** حدِّد الاختلافات بين هياكل البيانات الثابتة والمتغيرة.

| هياكل البيانات الثابتة | هياكل البيانات المتغيرة |
|---|---|
| | |

**3** اكتب مثالين لاستخدامات القوائم المترابطة.

_______________________________________________

_______________________________________________

_______________________________________________

_______________________________________________

_______________________________________________

**4** لديك مُكدّس به ست مساحات فارغة.

- ستُضيف الحروف الآتية C وE وB وA وD في المواقع من 0 إلى 4.
- املأ المُكدّس الذي يُشير إلى موقع المؤشر الأعلى.
- نَفِّذ العمليات التالية:

pop ← push K ← push X ← pop ← pop ←

اظهر المُخرَج النهائي بعد تنفيذ العمليات السابقة للإشارة إلى موقع المؤشر العلوي.

اكتب البرنامج الذي يُنشئ المُكدّس الموضّح بالأعلى، ثم نَفِّذ العمليات المذكورة أعلاه باستخدام مكتبة الطابور القياسية.

| المُكدّس | |
|---|---|
| 5 | |
| 4 | |
| 3 | |
| 2 | |
| 1 | |
| 0 | |

| المُخرَج النهائي | |
|---|---|
| 5 | |
| 4 | |
| 3 | |
| 2 | |
| 1 | |
| 0 | |

**5** لديك التسلسل الرقمي الآتي: 4 و8 و2 و5 و9 و13.

- ما العملية المُستخدَمة لإضافة العناصر الموضَّحة بالأعلى إلى الطابور؟

- أكمل الطابور بعد إضافة العناصر.

| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| | | | | | |

- ما العملية المُستخدَمة لحذف العناصر من الطابور؟

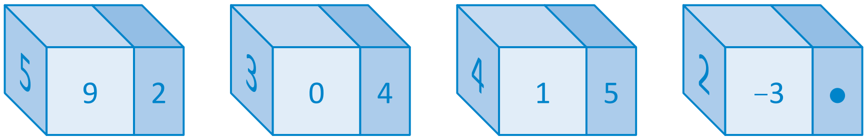
- كم مرة يجب تنفيذ العملية الموضَّحة بالأعلى لحذف العنصر الذي قيمته 5؟

- أكتب المقطع البرمجي بلغة البايثون لإنشاء الطابور السابق.

وزارة التعليم
Ministry of Education
2025 - 1447

**6** باستخدام العُقد التالية ارسِّم القائمة المترابطة، ثم اكتب القيم في القائمة بالترتيب السليم.

الرأس = 3




**7** أنشئ قائمة تضُم الأرقام التالية: 5 و20 و45 و8 و1.

- ارسم العُقد في القائمة المترابطة.

- صِفْ عملية إضافة الرقم 7 بعد الرقم 45.

- ارسم القائمة الجديدة.

- صِفْ العملية المطلوبة لحذف العُقدة الثانية من القائمة.

- ارسم القائمة المترابطة النهائية.

وزارة التعليم
Ministry of Education
2025 - 1447

رابط الدرس الرقمي

www.ien.edu.sa

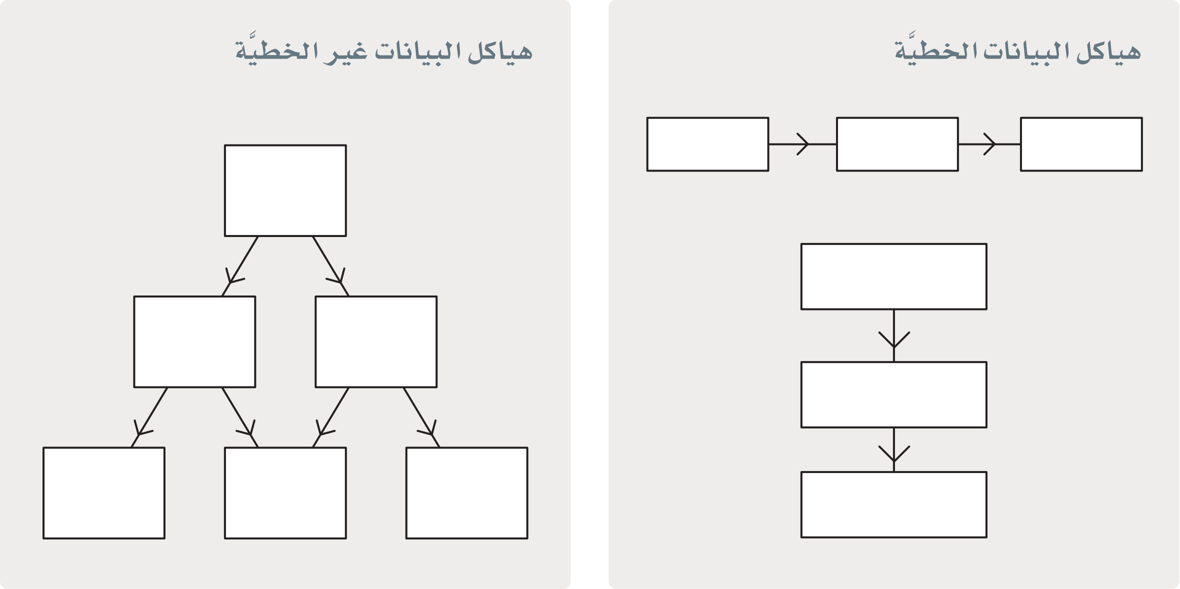
الدرس الثالث

# هياكل البيانات غير الخطيَّة

في الدرس السابق تعلّمت بعض هياكل البيانات الخطيَّة، وفيها كل عنصر يَتبع العنصر السابق له بطريقة خطيَّة. هل يمكنك التفكير في حالة لا تسير فيها الأشياء بتسلسل خطيّ؟ على سبيل المثال، هل يمكن لعنصر واحد أن يتبعه أكثر من عنصر؟

## هياكل البيانات غير الخطيَّة Non-Linear Data Structures

هي نوع من هياكل البيانات يتميز بإمكانية ربط عنصر بأكثر من عنصر واحد في الوقت نفسه. ومن الأمثلة التوضيحية على هياكل البيانات غير الخطيَّة: الأشجار ومُخطَّطات البيانات. الشكل 1.33 يوضِّح هياكل البيانات الخطيَّة وهياكل البيانات غير الخطيَّة.




شكل 1.33: الرسم التوضيحي لهياكل البيانات الخطيَّة وغير الخطيَّة

**جدول 1.9: الفرق بين هياكل البيانات الخطيَّة وغير الخطيَّة**

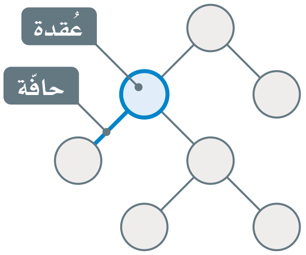
| هياكل البيانات الخطيَّة | هياكل البيانات غير الخطيَّة |
|---|---|
| تُرتَّب عناصر البيانات في ترتيب خطي يرتبط فيه كل عنصر بالعنصرين السابق والتالي له. | يمكن ربط عناصر البيانات بالعديد من العناصر الأخرى. |
| تُستعَرض عناصر البيانات في مسار واحد. | لا تُستعَرض عناصر البيانات في مسار واحد. |
| سهل التنفيذ. | معقّد التنفيذ. |



وزارة التعليم
Ministry of Education
2025 - 1447

## الأشجار Trees

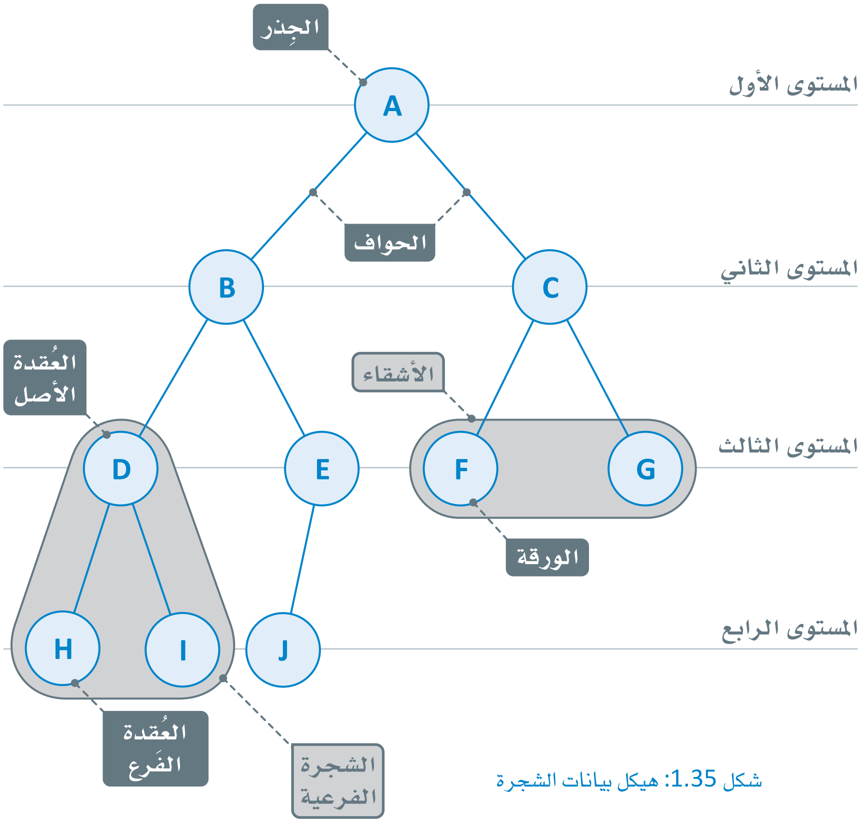
الأشجار هي نوع من هياكل البيانات غير الخطيَّة، وتتكون الشجرة من مجموعة من العُقد المُرتَّبة في ترتيب هرمي. ترتبط كل عُقدة بواحدة أو أكثر من العُقد، وترتبط العُقد مع الحواف في نموذج علاقة يربط بين **الأصل** (Parent) و**الفَرع** (Child). تُستخدَم الأشجار في العديد من مجالات علوم الحاسب، بما في ذلك أنظمة التشغيل، والرسوميات، وأنظمة قواعد البيانات، والألعاب، والذكاء الاصطناعي، وشبكات الحاسب.



شكل 1.34: العلاقات في الشجرة

### مُصطلحات تقنية الشجرة المُستخدمة في هيكل بيانات الشجرة
### Tree Terminology Used in the Tree Data Structure

- **الجِذر** (Root): العُقدة الأولى والوحيدة في الشجرة التي ليس لها أصل وتأتي في المستوى الأول من الشجرة، مثل: العُقدة A في الشكل 1.35.
- **الفَرع** (Child): العُقدة المرتبطة مباشرةً بِعُقدة في المستوى الأعلى، مثل: العُقدة H هي فَرع العُقدة D، والعُقدتان B وC هما فَرعا العُقدة A.
- **الأصل** (Parent): العُقدة التي لها فَرع أو أكثر في المستوى الأقل، مثل: العُقدة B هي أصل العُقدتين D وE.
- **الورقة** (Leaf): العُقدة التي ليس لها أي عُقدة فرعية، مثل: الورقة F.
- **الأشقاء** (Siblings): كل العُقد الفرعية التي تنبثق من الأصل نفسه، مثل:العُقدتان D وE شقيقتان.
- **الحواف** (Edges): الروابط التي تصل بين العُقد والشجرة.
- **الشجرة الفرعية** (Sub-Tree): الشجيرات التي توجد داخل الشجرة الأكبر حجمًا، مثل:الشجرة التي بها العُقدة D هي الأصل والعُقدتان H وI هما الفَرعان.



شكل 1.35: هيكل بيانات الشجرة

**الشجرة (Tree):**

الشجرة هي نوع من هياكل البيانات غير الخطيَّة، وتتكون من مجموعة من العُقد المُرتَّبة في ترتيب هرمي.

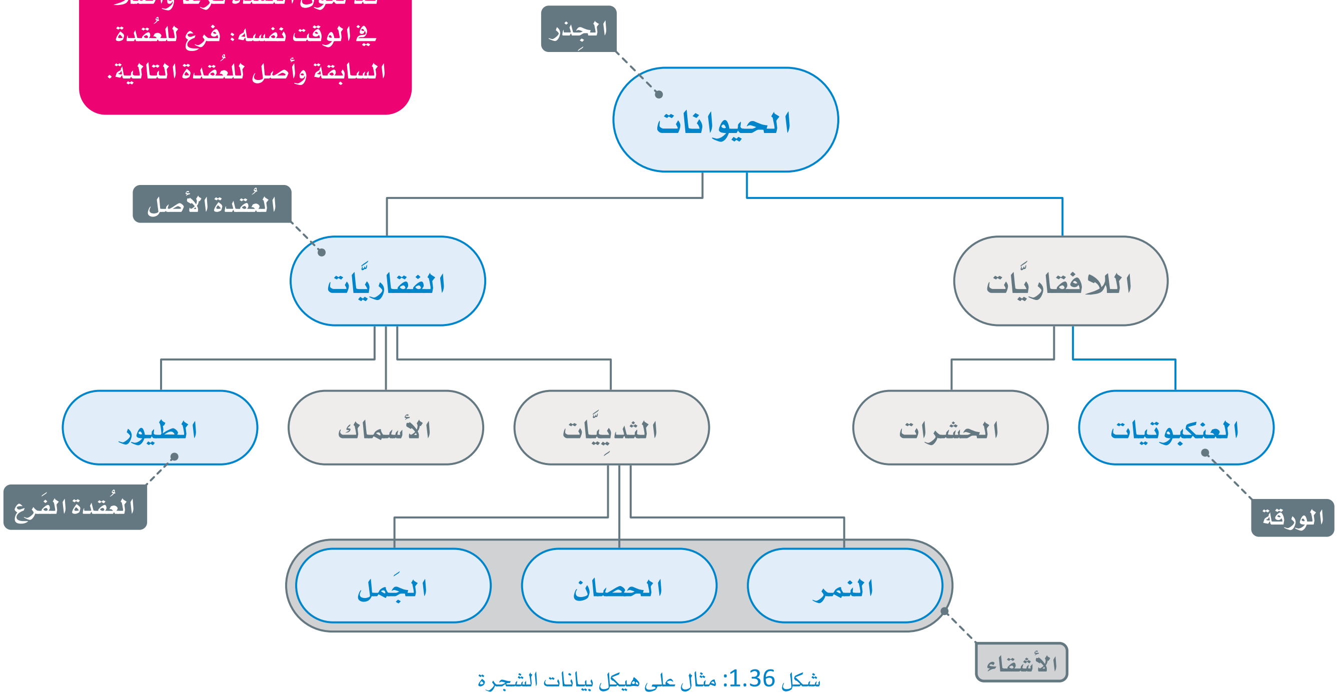
**الحافة (Edge):**

الحافة تصل بين عُقد هيكل بيانات الشجرة.

قد يكون لديك شجرة بسيطة تتكون من عُقدة واحدة. تكون هذه العُقدة في الوقت نفسه جذر هذه الشجرة البسيطة؛ لأنّها ليس لها أصل.

وفيما يلي مثال على هيكل بيانات الشجرة:

قد تكون العُقدة فرعًا وأصلًا في الوقت نفسه: فرع للعُقدة السابقة وأصل للعُقدة التالية.



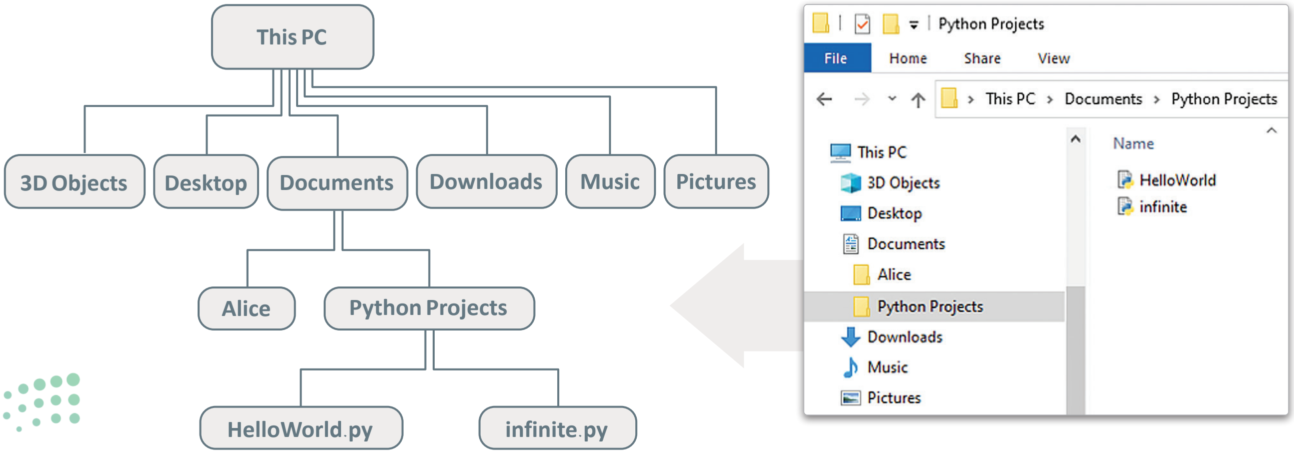

شكل 1.36: مثال على هيكل بيانات الشجرة

## خصائص هيكل بيانات الشجرة Tree Data Structure Features

- يُستخدم لتمثيل المُخطَّط الهرمي.
- يتميّز بالمرونة، فمن السهل إضافة عنصر من الشجرة أو حذفه.
- سهولة البحث عن العناصر فيه.
- يعكس العلاقات الهيكلية بين البيانات.

### مثال

تنظيم الملفات في نظام التشغيل هو مثال عملي على الشجرة. كما يتضح في الشكل 1.37، يوجد داخل مجلد Documents (المستندات) مُجلد آخر اسمه Python Projects (مشروعات البايثون) يحتوي على ملفين آخرين.

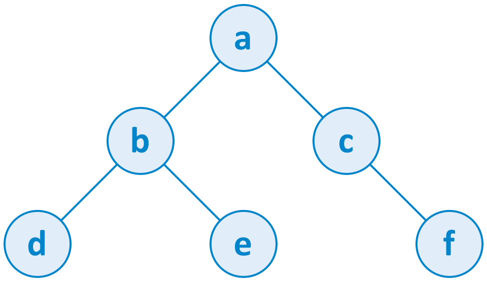



شكل 1.37: تنظيم الملفات في نظام التشغيل

## هيكل بيانات الشجرة في لغة البايثون

### Tree Data Structure in Python

لا تُوفِر لغة البايثون نوعًا محددًا مسبقًا من البيانات لهيكل بيانات الشجرة. ومع ذلك، تُصمَّم الأشجار من القوائم والقواميس بسهولة. يوضِّح الشكل 1.38 تطبيقًا بسيطًا للشجرة باستخدام القاموس.



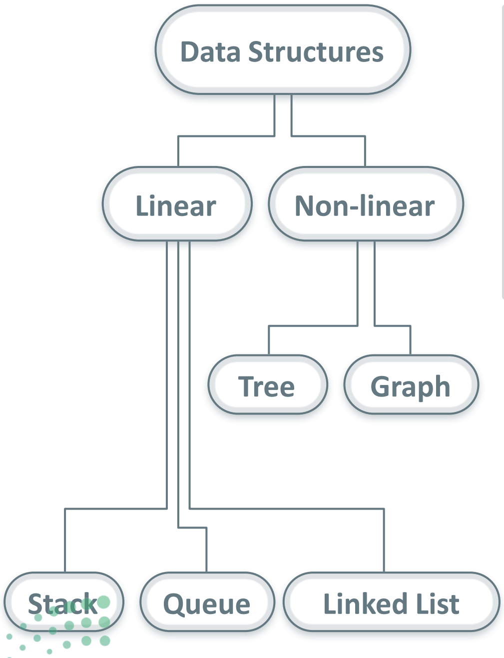
شكل 1.38: شجرة قاموس البايثون

في هذا المثال، ستُنشئ شجرة باستخدام قاموس البايثون. ستمثّل عُقد الشجرةِ مفاتيحَ القاموس، وستكون القيمة المقابلة لكل مفتاح هي قائمة تحتوي على العُقد المُتصلة بحافة مباشرة من هذه العُقدة.

```
myTree = {
    "a": ["b", "c"], # node
    "b": ["d", "e"],
    "c": [None, "f"],
    "d": [None, None],
    "e": [None, None],
    "f": [None, None],
}
print(myTree)
```

```
{'a': ['b', 'c'], 'b': ['d', 'e'], 'c': [None, 'f'],
'd': [None, None], 'e': [None, None], 'f': [None, None]}
```

في المثال التالي ستُنشئ شجرة مثل تلك الموضحة في الشكل 1.39:



شكل 1.39: شجرة هياكل البيانات

الأصل
الفَرع

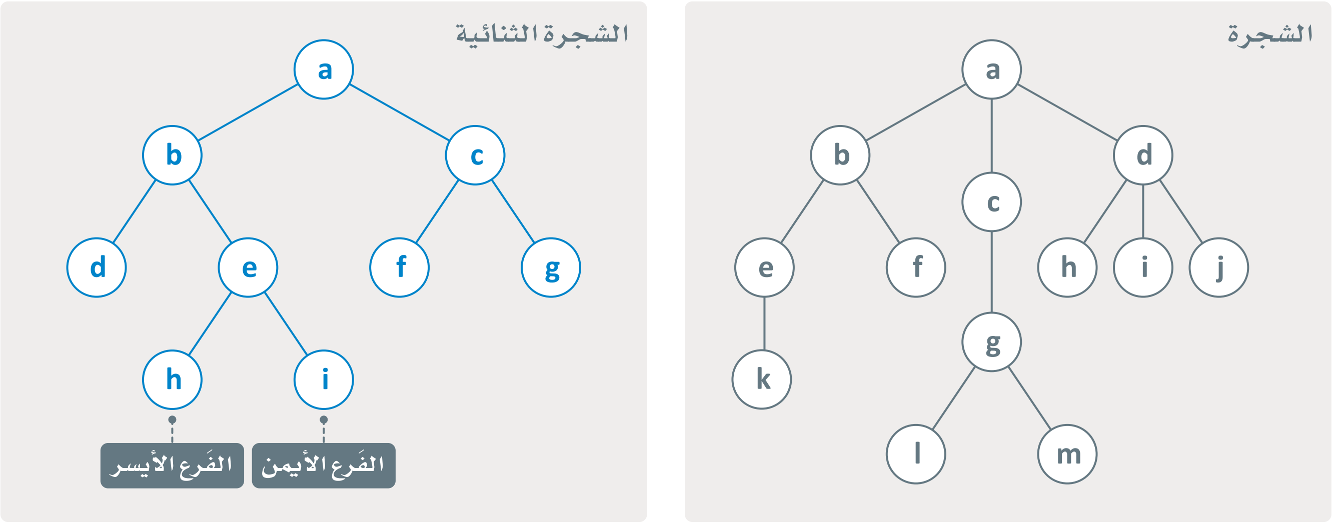
```
myTree = {"Data Structures":["Linear","Non-linear"],
          "Linear":["Stack","Queue","Linked List"],
          "Non-linear":["Tree", "Graph"]}

for parent in myTree:
    print(parent, "has",len(myTree[parent]),"nodes" )
    for children in myTree[parent]:
        print(" ",children)
```

```
Data structures has 2 nodes
  Linear
  Non-linear
Linear has 3 nodes
  Stack
  Queue
  Linked List
Non-linear has 2 nodes
  Tree
  Graph
```

وزارة التعليم
Ministry of Education
2025 - 1447

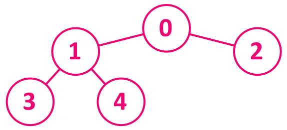
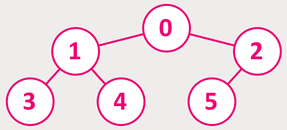
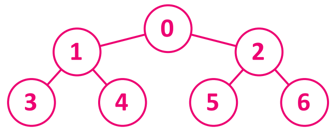
## الشجرة الثنائية Binary Tree

الشجرة الثنائية هي نوع خاص من الأشجار، يكون لكلّ عُقدةٍ فيها فَرعان على الأكثر؛ الفَرع الأيمن والفَرع الأيسر. الشكل 1.40 يَعرض مثالًا يوضِّح الشجرة والشجرة الثنائية.




شكل 1.40: الشجرة والشجرة الثنائية

**جدول 1.10: أنواع هياكل بيانات الشجرة الثنائية**

| النوع | الوصف | رسم توضيحي للهيكل |
|---|---|---|
| **الشجرة الثنائية التّامة** (Full Binary Tree) | يكون لكلّ عُقدة إمّا 0 أو 2 من الفروع (Children) بخلاف الأوراق (Leaves). |  |
| **الشجرة الثنائية الكاملة** (Complete Binary Tree) | يكون كلّ مستوًى من مستويات الشجرة ممتلئًا بالكامل، ربما باستثناء المستوى الأخير، حيث تكون كل العُقد فيه مملوءة من اليسار إلى اليمين. |  |
| **الشجرة الثنائية المثالية** (Perfect Binary Tree) | يكون لكل العُقد الداخلية فَرعان وتكون كل الأوراق عند المستوى نفسه. |  |

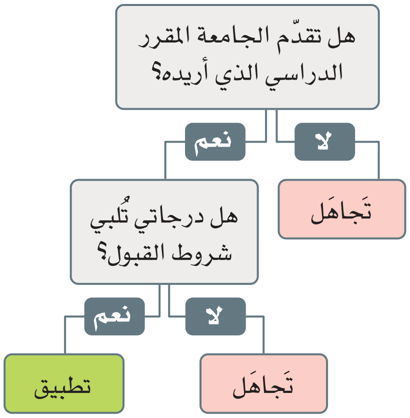
### أمثلة على تطبيقات هياكل بيانات الشجرة Examples of Applications of Tree Data Structures:

- تخزين البيانات الهرمية مثل: هياكل المجلدات.
- تعريف البيانات في **لغة ترميز النص التشعبي** (HTML).
- تنفيذ الفهرسة في قواعد البيانات.

## شجرة القرار Decision Tree

عبارة القرار if a: else b هي واحدة من العبارات الأكثر استخدامًا في لغة البايثون. ومن خلال تداخل وتجميع هذه العبارات، يمكنك تصميم شجرة القرار.

تُستخدَم أشجار القرار في الذكاء الاصطناعي من خلال إحدى تقنيات تعلُّم الآلة وتُعرف باسم: **تعلُّم شجرة القرار** (Decision Tree Learning). العُقد الأخيرة في هذه التقنية تُسمّى أيضًا الأوراق، وتحتوي على الحلول المُحتملة للمشكلة. كل عُقدة باستثناء الأوراق ترتبط بحالة منطقية يتفرع منها احتمالا الإجابة بنعم أو لا. أشجار القرار تُعدُّ سهلة الفهم، والاستخدام، والتصوير، ويسهل التحقق منها. على سبيل المثال، الشكل 1.41 يوضِّح شجرة القرار التي تُحدِّد ما إذا كنت ستَتقدَّم بطلب الالتحاق بجامعة مُحدَّدة أم لا بناءً على معيارين: المقررات الدراسية التي تُدرَّس في الجامعة، واستيفاء متطلبات القبول.



شكل 1.41: مثال على شجرة القرار

## المُخطَّطات Graphs

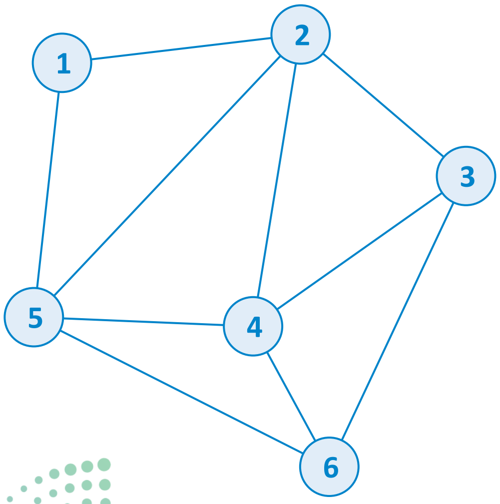
السمَة الأكثر أهمية لهياكل البيانات غير الخطيَّة هي أنّ البيانات الخاصة بها لا تتبّع أي نوع من أنواع التّسلسل، وذلك على خلاف المصفوفات والقوائم المترابطة، كما يمكن ربط عَناصرها بأكثر من عنصر وحيد. **الشجرة الجذرية** (Rooted Tree) تبدأ بعُقدة جذرية يمكن ربطها بالعُقد الأخرى. تَتّبع الأشجار قواعد محددة: وهي أن تكون عُقد الشجرة متصلة، وأن تكون الشجرة خالية من **الحلقات** (Loops) و**الحلقات الذاتية** (Self Loops)، كما أن لبعض أنواع الأشجار قواعدها الخاصة (جدول 1.10)، مثلما في حالة الأشجار الثنائية. ولكن ماذا سيحدث إذا لم تَتَّبع قواعد الأشجار؟ في هذه الحالة أنت لا تتحدث عن الأشجار، بل عن نوع جديد من هياكل البيانات المُتغيِّرة التي تُسمى المُخطَّطات. في الحقيقة، الأشجار هي نوع من المُخطَّطات حيث أن المُخطَّط هو الشكل العام لهيكل البيانات، بمعنى أن كل هياكل البيانات السابقة يمكن اعتبارها حالات خاصة من المُخطَّطات. الشكل 1.42 يعرض مُخطَّطًا به ست عُقد وعشر حواف.

**المُخطَّط (Graph):**
المُخطَّط هو هيكل البيانات المُكوَّن من مجموعة من العُقد ومجموعة من الخطوط التي تصل بين جميع العُقد، أو بعضها.

كل الأشجار مُخطَّطات، ولكن ليست كل المُخطَّطات أشجارًا.

**جدول 1.11: الفرق بين الأشجار والمُخطَّطات**

| الأشجار | المُخطَّطات |
|---|---|
| تشكّل العُقد المتّصلة فيها نموذجًا هرميًّا. | تشكّل العُقد المتّصلة فيها نموذجًا شبكيًّا. |
| في الأشجار الجذرية توجد عُقدة فريدة تُسمى الجِذر. | لا توجد فيها عقدة فريدة أو جذرية. |
| ترتبط العُقد في صورة علاقة بين الأصل والفرع. | لا تنطبق علاقة الأصل والفرع بين العُقد. |
| تتميز ببساطة التركيب. | تركيب المُخطَّطات أكثر تعقيدًا. |
| لا يُسمح فيها بالحلقات. | قد تحتوي على الحلقات. |

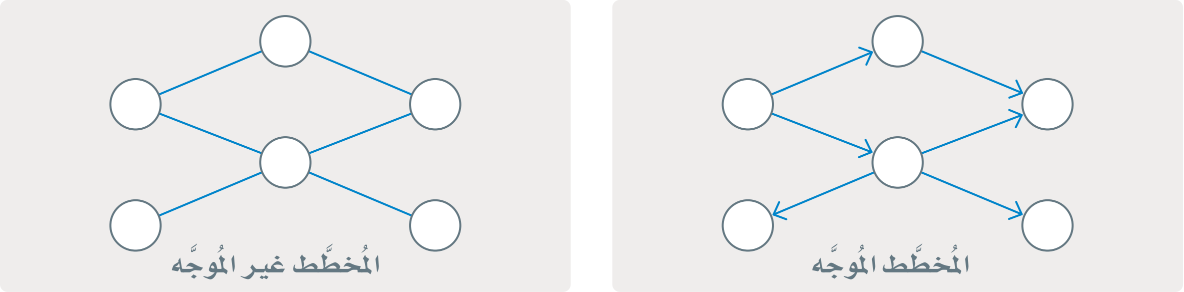


شكل 1.42: مثال على مُخطَّط به ست عُقد وعشر حواف

### أنواع المُخطَّطات Types of Graphs

- **المُخطَّط المُوجَّه** (Directed Graph): ترتبط العُقد بالحواف الموجهة في المُخطَّط المُوجَّه، بحيث يكون للحافة اتجاه واحد.
- **المُخطَّط غير المُوجَّه** (Undirected graphs): لا تحتوي الوصلات على اتجاه في المُخطَّط غير المُوجَّه، وهذا يعني أن الحواف تشير إلى علاقة ثنائية الاتجاه يمكن من خلالها عرض البيانات في كلا الاتجاهين.

الشكل 1.43 يعرض مُخطَّطًا موجَّهًا، ومُخطَّطًا غير مُوجَّه يتكونان من ست عُقد وست حواف.




شكل 1.43: المُخطَّط المُوجَّه والمُخطَّط غير المُوجَّه

## المُخطَّطات في الحياة اليومية Graphs in Everyday Life

### شبكة الويب العالمية World Wide Web

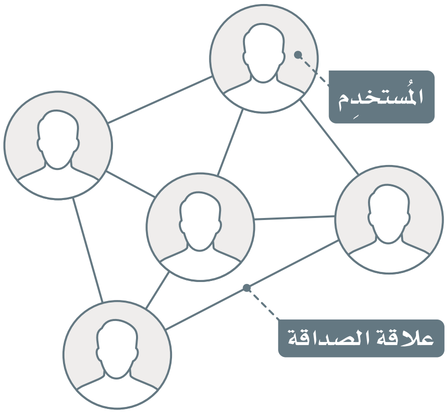
تُعدُّ شبكة الويب العالمية من أبرز الأمثلة للمُخطَّطات، ويمكن اعتبارها بمثابة أحد أنواع المُخطَّطات المُوجَّهة حيث تُمثِّل **الرؤوس** (Vertices) صفحات الويب، وتُمثِّل الارتباطات التشعبية الحواف المُوجَّهة. **تنقيب بُنيَة الويب** (Web Structure Mining) هو اكتشاف المعرفة المُفيدة من هيكل شبكة الويب المُمَثَلة من خلال الارتباطات التشعبية، ويمكن أن يمثّل هيكل المُخطَّط الارتباطات التشعبية والعلاقات التي تُنشئها بين صفحات الويب المختلفة. يعرض الشكل 1.44 رسمًا توضيحيًا لشبكة الويب العالمية. باستخدام هذه المُخطَّطات يُمكنك حساب الأهمية النسبية لصفحات الويب.

شكل 1.44: شبكة الويب العالمية

يَستخدِم **مُحرِّك البحث قوقل** (Google Search Engine) منهجية مماثلة لتحديد الأهمية النسبية لصفحات الويب ومن ثمَّ ترتيب نتائج البحث حسب أهميتها. الخوارزمية المُستخدَمة بواسطة قوقل هي خوارزمية **تصنيف الصفحة** أو **بيج رانك** (PageRank) التي ابتكرها مؤسّسو قوقل.


وزارة التعليم
Ministry of Education
2025 - 1447

## فيسبوك Facebook

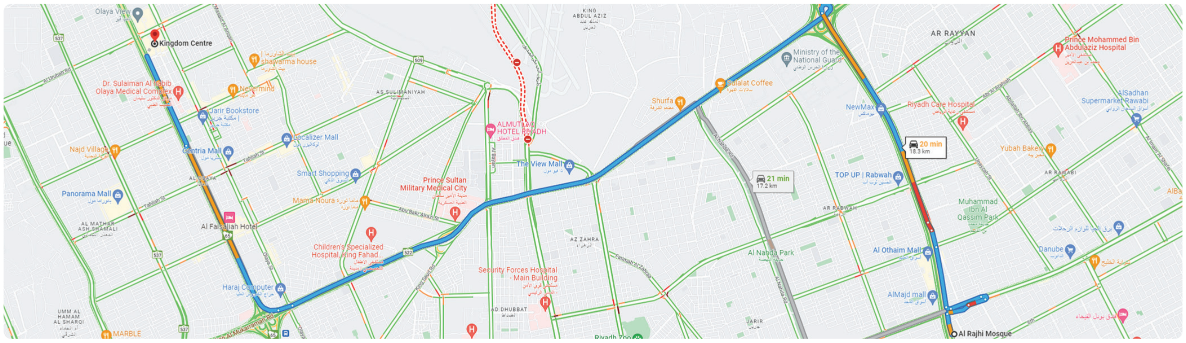



شكل 1.45: مُخطَّط فيسبوك غير المُوجَّه

فيسبوك هو مثال آخر على المُخطَّطات غير المُوجَّهة. يظهر بالشكل 1.45 العُقد التي تُمثِّل مُستخدِمي فيسبوك، بينما تُمثِّل الحواف علاقات الصداقة. عندما تريد إضافة صديق، يجب عليه قبول طلب الصداقة؛ ولن يكون ذلك الشخص صديقك على الشبكة دون قبول طلب الصداقة. العلاقة هنا بين اثنين من المُستخدِمين (عُقدتين) هي علاقة ثنائية الاتجاه. تُستخدَم خوارزمية مقترحات الأصدقاء في فيسبوك نظرية المُخطَّطات. تَدرس تحليلات الشبكات الاجتماعية العلاقات الاجتماعية باستخدام نظرية المُخطَّطات أو الشبكات من علوم الحاسب.

## خرائط قوقل Google Maps

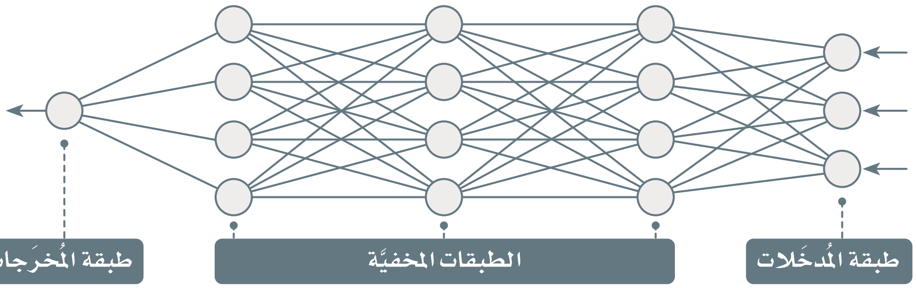
يستخدِم تطبيق خرائط قوقل وكل التطبيقات المُشابهة له المُخطَّطات لعرض أنظمة النقل والمواصلات لحساب المسار الأقصر بين موقعين. تَستخدِم هذه التطبيقات المُخطَّطات التي تحتوي على عدد كبير جدًا من العُقد والحواف التي لا يُمكن تمييزها بالعين المُجردة.




شكل 1.46: خرائط قوقل

## الشبكة العصبية Neural Network

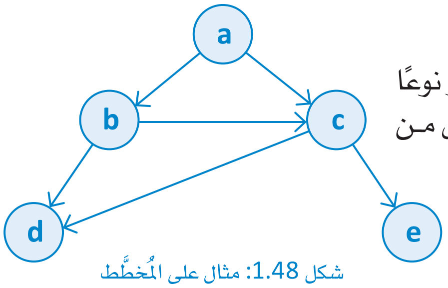
الشبكة العصبية هي نوع مُخطَّط تعلُّم الآلة الذي يُحاكي الدماغ البشري. الشبكات العصبية يُمكن أن تكون شبكات مُوجَّهة أو غير مُوجَّهة وفقًا للغرض من التعلُّم، وتتكوّن هذه الشّبكات من الخلايا العصبية والأوزان المُوزعة في الطبقات المختلفة. تُمثَّل الخلايا العصبية بالعُقد، بينما تُمثَّل الأوزان بالحواف. يتم حساب تدفقات الإشارة وتحسينها في جميع أنحاء بُنية الشبكات العصبية لتقليل الخطأ. تُستخدَم الشبكات العصبية في العديد من التطبيقات الذكية مثل: الترجمة الآلية، وتصنيف الصور، وتحديد الكائنات، والتعرّف عليها. الشكل 1.47 يوضِّح مثالًا على هيكل الشبكات العصبية.




شكل 1.47: هيكل الشبكات العصبية

## المُخطَّطات في لغة البايثون Graphs in Python

لا تُوفِر لغة البايثون نوعًا محددًا مسبقًا من البيانات للأشجار، كما أنّها لا تُوفِر نوعًا محددًا مسبقًا من البيانات للمُخطَّطات، (تذكر أن الأشجار هي نوع خاص من المُخطَّطات). ومع ذلك، يُمكن بناء المُخطَّطات باستخدام القوائم والقواميس.



شكل 1.48: مثال على المُخطَّط

في المثال التالي، ستقوم بتنفيذ التالي:

1. إنشاء مُخطَّط مُوجَّه مثل المُوضح بالشكل 1.48.
2. إنشاء دالة لإضافة عُقدة إلى المُخطَّط.
3. إنشاء كائن يحتوي على كل مسارات المُخطَّط.

```
myGraph = { "a" : ["b","c"],
            "b" : ["c", "d"],
            "c" : ["d", "e"],
            "d" : [],
            "e" : [],
            }
print(myGraph)
```

```
{'a': ['b', 'c'], 'b': ['c', 'd'], 'c': ['d', 'e'],
 'd': [], 'e': []}
```

وسيتولّى البرنامج الرئيس:

1. إنشاء المُخطَّط.
2. طباعة المُخطَّط.
3. استدعاء دالة الإضافة.
4. طباعة كل مسارات المُخطَّط.

ستَستخدِم القاموس الذي تُمثِّل مفاتيحه العُقد بالمُخطَّط. تكون القيمة المقابِلة لكل مفتاح هي قائمة تحتوي على العُقد المتصلة بحافة مباشرة من هذه العُقدة.

```
# function for adding an edge to a graph
def addEdge(graph,u,v):
    graph[u].append(v)

# function for generating the edges of a graph
def generate_edges(graph):
    edges = []

    # for each node in graph
    for node in graph:
```


وزارة التعليم
Ministry of Education
2025 - 1447

```
        # for each neighbouring node of a single node
        for neighbour in graph[node]:

            # if edge exists then append to the list
            edges.append((node, neighbour))
    return edges

# main program
# initialisation of graph as dictionary
myGraph = {"a" : ["b","c"],
           "b" : ["c", "d"],
           "c" : ["d", "e"],
           "d" : [],
           "e" : [],
          }

# print the graph contents
print("The graph contents")
print(generate_edges(myGraph))

# add more edges to the graph
addEdge(myGraph,'a','e')
addEdge(myGraph,'c','f')

# print the graph after adding new edges
print("The new graph after adding new edges")
print(generate_edges(myGraph))
```

```
The graph contents
[('a', 'b'), ('a', 'c'), ('b', 'c'), ('b', 'd'), ('c', 'd'), ('c', 'e')]
The new graph after adding new edges
[('a', 'b'), ('a', 'c'), ('a', 'e'), ('b', 'c'), ('b', 'd'), ('c', 'd'),
('c', 'e'), ('c', 'f')]
```

# تمرينات

**1**

| حدِّد الجملة الصحيحة والجملة الخاطئة فيما يلي: | صحيحة | خاطئة |
|---|---|---|
| 1. يمكن ربط العنصر في هياكل البيانات غير الخطيَّة بأكثر من عنصر واحد. | ○ | ○ |
| 2. تنفيذ هياكل البيانات الخطيَّة يكون أكثر تعقيدًا من تنفيذ هياكل البيانات غير الخطيَّة. | ○ | ○ |
| 3. الأوراق في تعلُّم شجرة القرار تحتوي على حلول المشكلة. | ○ | ○ |
| 4. تَحسِب خوارزمية قوقل تصنيف الصفحة (PageRank) الأهمية النسبية لصفحة ويب على شبكة الويب العالمية. | ○ | ○ |
| 5. الشبكات العصبية هي نوع المُخطَّطات المُستخدَم لتصوير المشكلات الأخرى. | ○ | ○ |

**2** وضِّح الاختلافات بين الأشجار والمُخطَّطات.

| الأشجار | المُخطَّطات |
|---|---|
| | |

**3** صِفْ كيف تُستخدَم خوارزميات المُخطَّطات في التطبيقات التجارية.

______________________________

______________________________

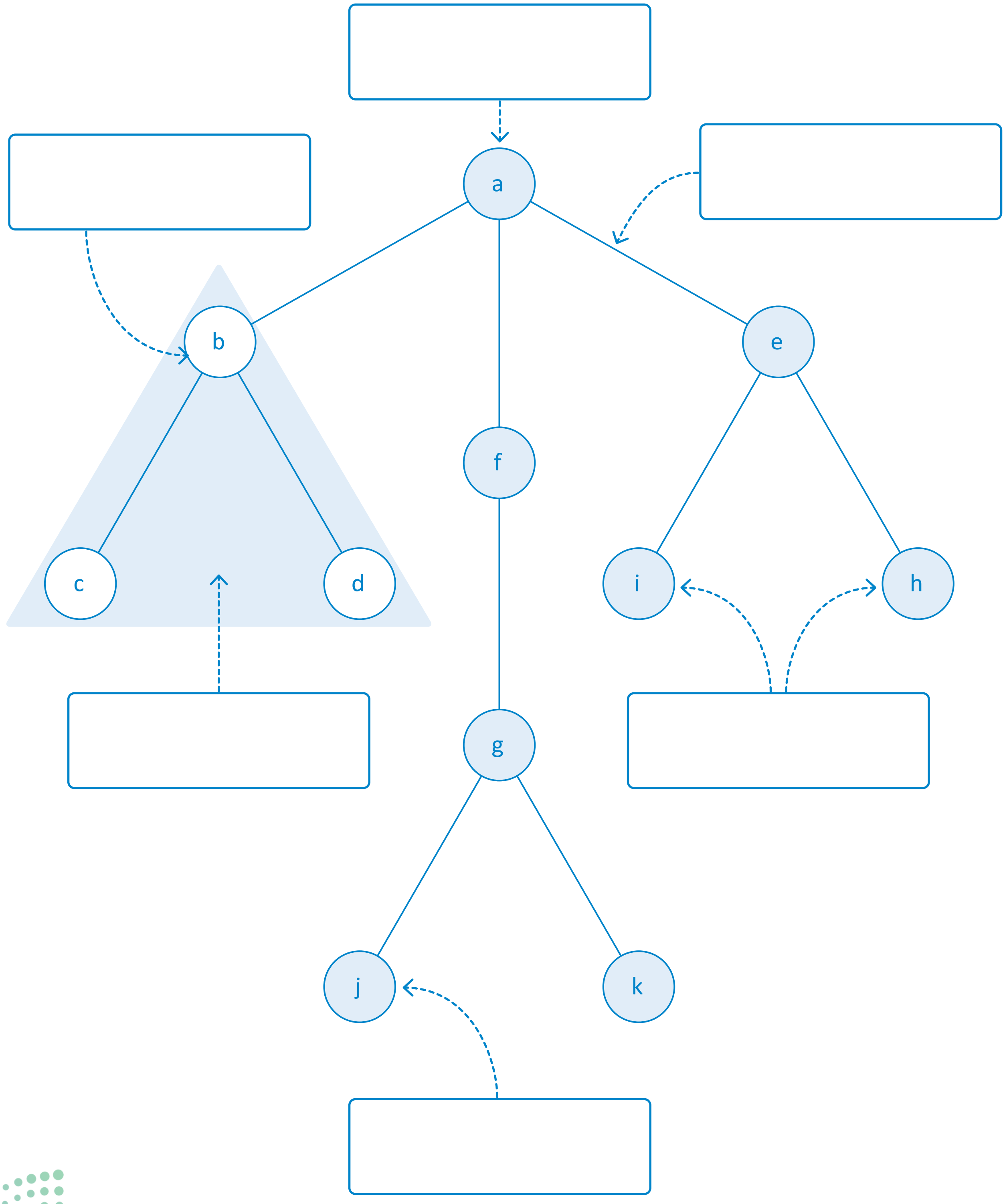
______________________________

______________________________



وزارة التعليم
Ministry of Education
2025 - 1447

**4** املأ الفراغات بالأسماء الصحيحة لأجزاء الشجرة.

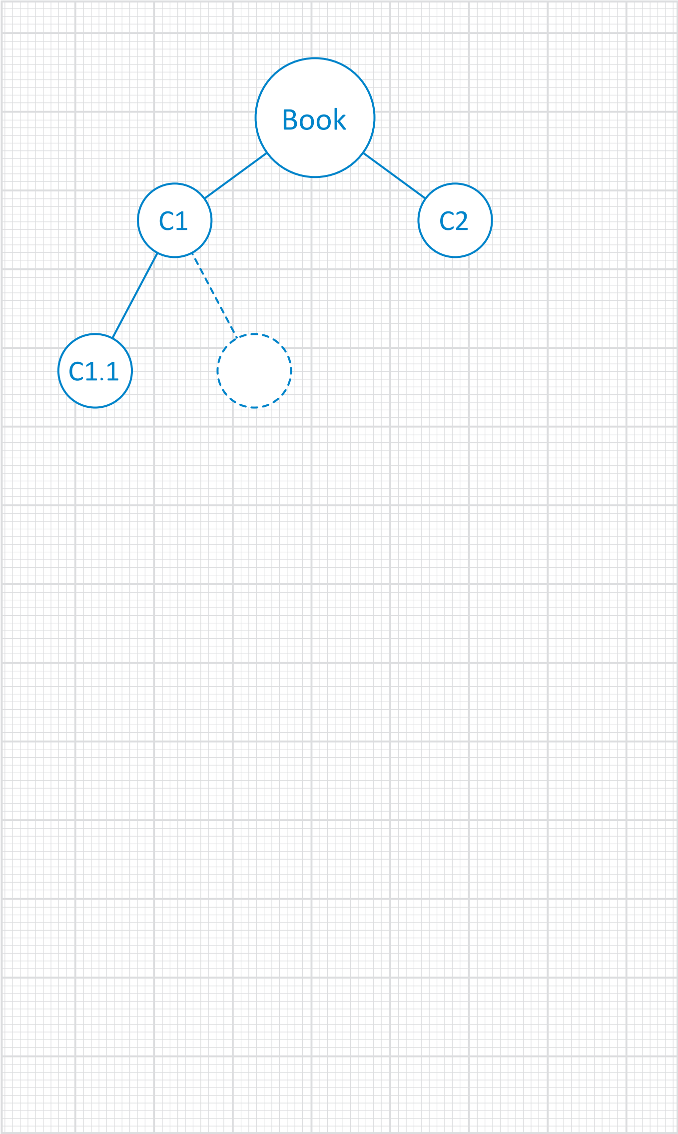


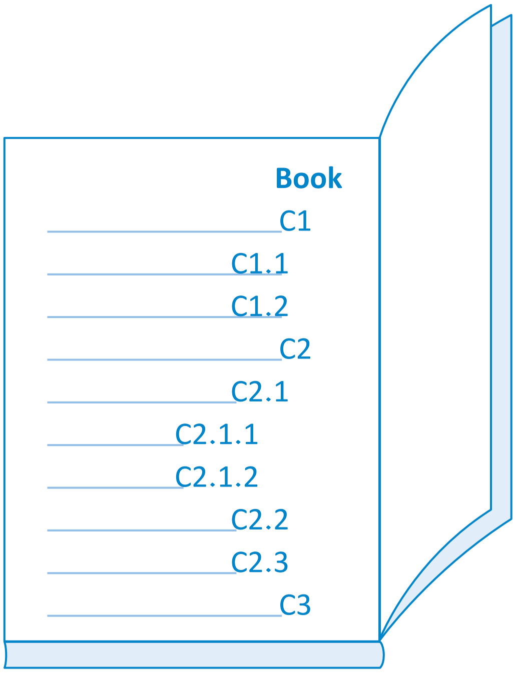



وزارة التعليم
Ministry of Education
2025 - 1447

5 يظهر أمامك في الصورة التالية صفحة محتويات الكتاب.

- أكمل تمثيل الشجرة.




- هل هي شجرة ثنائية؟ عَلِّل إجابتك.


وزارة التعليم
Ministry of Education
2025 - 1447

**6** ارسم الشجرة الناتجة عن المعطيات التالية:

- العُقدة A لها فَرعان B وC.
- العُقدتان D وE لهما الأصل نفسه وهو العُقدة B.
- العُقدتان F وG شقيقتان، ولهما الأصل نفسه وهو العُقدة C.
- العُقدة H لها عُقدتان فَرعيتان I وJ ولها عُقدة أصل F.

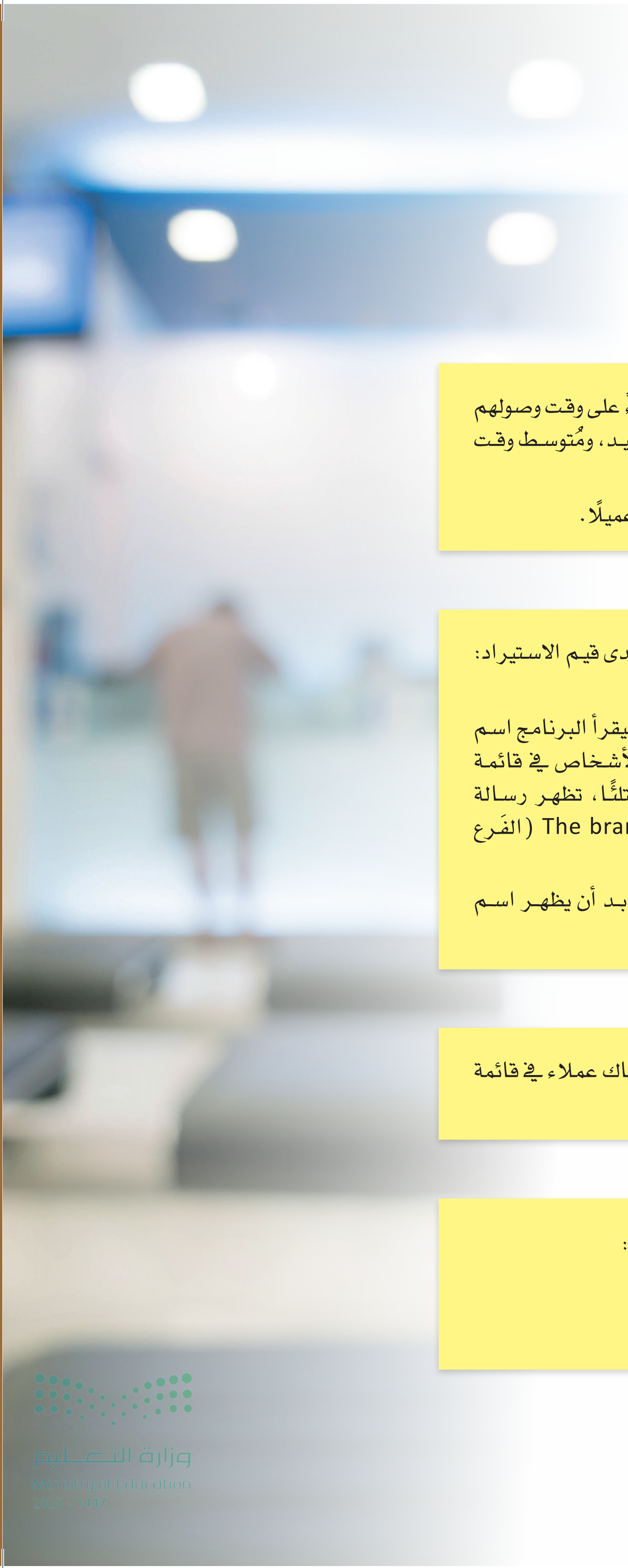
ما نوع الشجرة المرسومة في الأعلى؟

باستخدام القاموس في لغة البايثون اكتب البرنامج المناسب لتمثيل هذه الشجرة، ثم أضف العُقدة الأصل والعُقد الفرعية.



وزارة التعليم
Ministry of Education
2025 - 1447

# المشروع



تُقدَّم الخدمة للعملاء في أحد البنوك بناءً على وقت وصولهم إلى فرع البنك. يعمل بالبنك موظف وحيد، ومُتوسط وقت الخدمة لكل عميل هو دقيقتان.

لا يُسمح بأن يتجاوز الطابور في البنك 40 عميلًا.

1

أنشئ برنامجًا بلغة البايثون يستدعي إحدى قيم الاستيراد: ENTRY (دخول) أو NEXT (التالي).

- إن أدخلت القيمة ENTRY (دخول)، سيقرأ البرنامج اسم العميل وبعدها مباشرةً يُظهِر عدد الأشخاص في قائمة الانتظار أمامه. إن كان الطابور مُمتلئًا، تظهر رسالة The branch is full. Come another day (الفَرع مُمتلئ. الرجاء العودة في يوم آخر).
- إن أدخلت القيمة NEXT (التالي)، لابد أن يظهر اسم العميل التالي الذي ستُقدَّم له الخدمة.

2

كَرِّر العملية الموضحة أعلاه حتى لا يكون هناك عملاء في قائمة الانتظار.

3

في النهاية، سيَعرض البرنامج على الشاشة:

- عدد العملاء الذين قُدِّمت لهم الخدمة.
- متوسط وقت انتظار العميل.



وزارة التعليم
Ministry of Education
2025 - 1447

# ماذا تعلّمت

- مفهوم الذكاء الاصطناعي.
- تصنيف تطبيقات الذكاء الاصطناعي.
- تصنيف هياكل البيانات.
- تحديد الاختلافات بين هيكل بيانات المُكدّس وهيكل بيانات الطابور.
- تحديد الاختلافات بين هيكل بيانات القائمة وهيكل بيانات القائمة المترابطة.
- تحديد الاختلافات بين هيكل بيانات الشجرة وهيكل بيانات المُخطَّط.
- تطبيق هياكل البيانات المُعقَّدة باستخدام لغة برمجة البايثون.

## المصطلحات الرئيسة

| | |
|---|---|
| غير أوليّ | Non-Primitive |
| قيمة فارغة | Null |
| مؤشر | Pointer |
| حذف عنصر | Pop |
| أوليّ | Primitive |
| إضافة عنصر | Push |
| أخير | Rear |
| الجِذر | Root |
| أشقاء | Siblings |
| المُكدِّس | Stack |
| شجرة فرعية | Sub-Tree |
| أعلى | Top |
| غَيْض المُكدّس | Underflow |
| المُخطَّط غير الموجَّه | Undirected Graph |

| | |
|---|---|
| الشجرة الثنائية | Binary Tree |
| فَرع (ابن) | Child |
| هيكل البيانات | Data Structure |
| شجرة القرار | Decision Tree |
| حذف عنصر من الطابور | Dequeue |
| المُخطَّط المُوجّه | Directed Graph |
| متغير | Dynamic |
| الأمامي | Front |
| مُخطَّط | Graph |
| فهرس | Index |
| رأس | Head |
| ورقة | Leaf |
| خطي | Linear |
| قائمة مترابطة | Linked List |
| غير خطي | Non-Linear |

وزارة التعليم
Ministry of Education
2025 - 1447

# 2. خوارزميات الذكاء الاصطناعي

سيتعرف الطالب في هذه الوحدة على بعض الخوارزميات الأساسية المُستخدَمة في الذكاء الاصطناعي (AI). كما سيتعلّم كيف يُنشِئ نظام تشخيص طبي بسيط مُستنِد إلى القواعد بِطرائق برمجية مُتعددة ثم يقارن النتائج. وفي الختام سيتعلّم خوارزميات البحث وطرائق حل ألغاز المتاهة مع أخذ معايير معيّنة في الاعتبار.

## أهداف التعلُّم

بنهاية هذه الوحدة سيكون الطالب قادرًا على أن:

- يُنشئ مقطعًا برمجيًّا تكراريًّا.
- يُقارِن بين خوارزمية البحث بأولوية الاتساع وخوارزمية البحث بأولوية العُمق.
- يَصِف خوارزميات البحث وتطبيقاتها.
- يُقارِن بين خوارزميات البحث.
- يَصِف النظام القائم على القواعد.
- يُدرِّب نماذج الذكاء الاصطناعي حتى تتعلّم حل المشكلات المُعقدة.
- يُقيِّم نتائج المقطع البرمجي وكفاءة البرنامج الذي أنشأه.
- يُطوِّر البرامج لمحاكاة حلّ مشكلات الحياة الواقعية.
- يُقارِن بين خوارزميات البحث.

## الأدوات

- مفكّرة جوبيتر (Jupyter Notebook)

وزارة التعليم
Ministry of Education
2025 - 1447

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الأول

# الاستدعاء الذاتي

## تقسيم المُشكلة Dividing the Problem

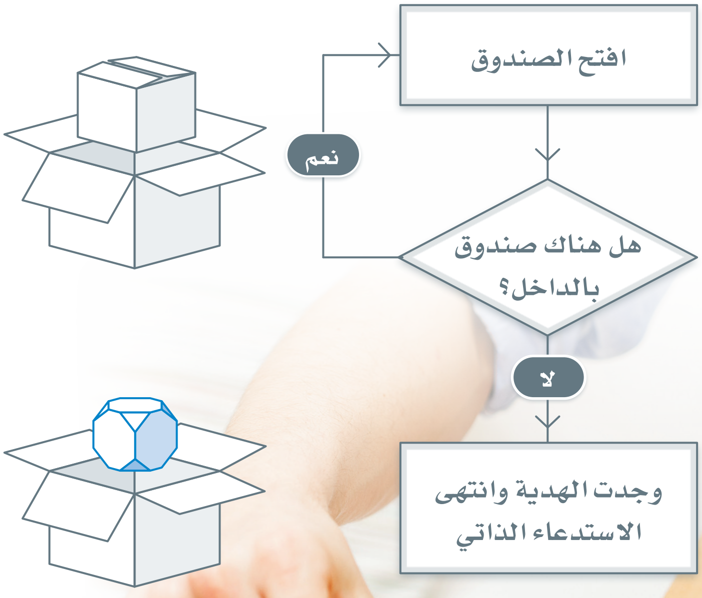
في هذا الدرس، ستتعلّم استخدام الدوال التكرارية لتبسيط البرنامج وزيادة كفاءته.

تخيّل أن والداكَ قد أحضرا لكَ هديّة، وكنت مُتلهفًا لمعرفتها، ولكن عندما فتحت الصندوق، وجدت صندوقًا جديدًا بداخله، وعندما فتحته، وجدتَ آخرَ بداخله، وهكذا حتى عجزت أن تعرف في أي صندوق توجد الهدية.

### الاستدعاء الذاتي Recursion

الاستدعاء الذاتي هو أحد طرائق حل المشكلات في علوم الحاسب، ويتم عن طريق تقسيم المشكلة إلى مجموعة من المشكلات الصغيرة المُشابهة للمشكلة الأصلية حتى يُمكنك استخدام الخوارزمية نفسها لحل تلك المشكلات. يُستخدَم الاستدعاء الذاتي بواسطة أنظمة التشغيل والتطبيقات الأخرى، كما تدعمه معظم لغات البرمجة.

يحدث الاستدعاء الذاتي عندما تتكرر التعليمات نفسها، ولكن مع بيانات مختلفة وأقل تعقيدًا.




شكل 2.1: مثال على الاستدعاء الذاتي

وزارة التعليم
Ministry of Education
2025 - 1447

لِتُلقِ نظرة على مثال لدالة تستدعي دالة أخرى.

```
def mySumGrade (gradesList):
    sumGrade=0
    l=len(gradesList)
    for i in range(l):
        sumGrade=sumGrade+gradesList[i]
    return sumGrade

def avgFunc (gradesList):
    s=mySumGrade(gradesList)
    l=len(gradesList)
    avg=s/l
    return avg

# program section
grades=[89,88,98,95]
averageGrade=avgFunc(grades)
print ("The average grade is: ",averageGrade)
```

استدعاء الدالة .mySumGrade

تستخدِم دالة ()len قائمة كمُعامِل مُدخَل، لحساب وتحديد عدد العناصر في القائمة.

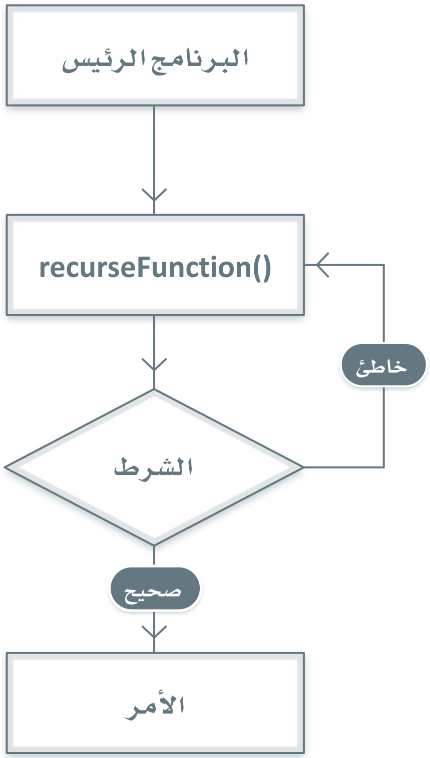
```
The average grade is: 92.5
```

## دالة الاستدعاء التكرارية Recursive Function

في بعض الحالات تستدعي الدّالةُ نفسَها وهذه الخاصية تُسمى **الاستدعاء التكراري** (Recursive Call).

يكون بِناء الجملة العام لدالة الاستدعاء التكرارية على النحو التالي:

```
# recursive function
def recurseFunction():
    if (condition): # base case
        statement
    else:
        #recursive call
        recurseFunction()

# main program
.......

# normal function call
recurseFunction()
.........
```



شكل 2.2: تمثيل الاستدعاء التكراري

**الاستدعاء التكراري هو عملية استدعاء الدالة لنفسها.**

تتكون دالة الاستدعاء التكرارية من حالتين:

### الحالة الأساسية Base Case

وفي هذه الحالة تتوقف الدالة عن استدعاء نفسها، ويتأكّد الوصول إلى هذه الحالة من خلال الأمر المشروط. بدون الحالة الأساسية، ستَتَكرَّر عملية الاستدعاء الذاتي إلى ما لا نهاية.

### حالة الاستدعاء التكرارية Recursive Case

وفي هذه الحالة تستدعي الدالةُ نفسَها عندما لا تُحقق شرط التوقف، وتظل الدالة في حالة الاستدعاء الذاتي حتى تصل إلى الحالة الأساسية.

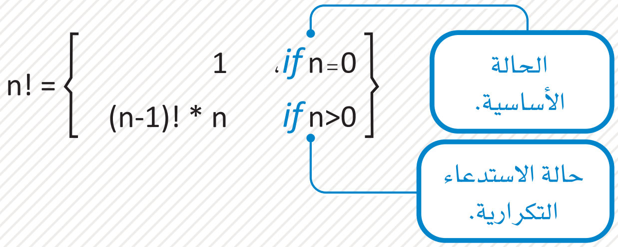
## أمثلة شائعة على الاستدعاء الذاتي Recursion Common Examples

أحد الأمثلة الأكثر شيوعًا على استخدام الاستدعاء الذاتي هو عملية حساب مضروب رقم مُعيّن. مضروب الرقم هو ناتج ضرب جميع الأعداد الطبيعية الأقل من أو تساوي ذلك الرقم. يُعبَّر عن المضروب بالرقم متبوعًا بالعلامة "!"، على سبيل المثال، مضروب الرقم 5 هو !5 ويساوي 1*2*3*4*5.

**جدول 2.1: مضروب الأرقام من 0 إلى 5**

| | | | |
|---|---|---|---|
| 0! | 0! = 1 | | |
| 1! | 1! = 1*1 = 1 | أو | 1! = 0! *1 |
| 2! | 2! = 2*1 = 2 | أو | 2! = 1! *2 |
| 3! | 3! = 3*2*1 = 6 | أو | 3! = 2! *3 |
| 4! | 4! = 4*3*2*1 = 24 | أو | 4! = 3! * 4 |
| 5! | 5!= 5*4*3*2*1 = 120 | أو | 5! = 4! * 5 |

ستلاحظ أن عملية حساب المضروب تستند إلى القاعدة أدناه:




شكل 2.3: قاعدة حساب المضروب

لإنشاء برنامج يقوم باحتساب مضروب العدد باستخدام حلقة التكرار for، اتّبع ما يلي:

```
# calculate the factorial of an integer using iteration

def factorialLoop(n):
    result = 1
    for i in range(2,n+1):
          result = result * i

    return result

# main program
num = int(input("Type a number: "))
f=factorialLoop(num)
print("The factorial of ", num, " is:", f)
```

```
Type a number: 3
The factorial of 3 is:6
```

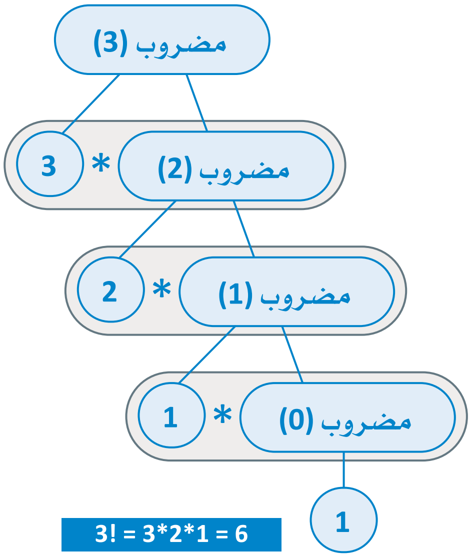
الآن احسِب مضروب العدد باستخدام دالة المضروب.

```
# calculate the factorial of an integer using a
# recursive function
def factorial(x):
    if x == 0:
        return 1
    else:
        return (x * factorial(x-1))

# main program
num = int(input("Type a number: "))
f=factorial(num)
print("The factorial of ", num, " is: ", f)
```

الحالة الأساسية.

حالة الاستدعاء التكرارية.

```
Type a number: 3
The factorial of 3 is: 6
```




شكل 2.4: شجرة الاستدعاء الذاتي

## جدول 2.2: مزايا الاستدعاء الذّاتي وعيوبه

| المزايا | العيوب |
|---|---|
| • تقلل دوال الاستدعاء التكرارية من عدد التعليمات في المقطع البرمجي. | • في بعض الأحيان، يَصعُب تَتبُّع منطق دوال الاستدعاء التكرارية. |
| • يمكن تقسيم المهمة إلى مجموعة من المشكلات الفرعية باستخدام الاستدعاء الذاتي. | • يتطلب الاستدعاء الذاتي مزيدًا من الذاكرة والوقت. |
| • في بعض الأحيان، يَسهُل استخدام الاستدعاء الذاتي لاستبدال التكرارات المُتداخلة. | • لا يسهل تحديد الحالات التي يمكن فيها استخدام دوال الاستدعاء التكرارية. |

## الاستدعاء الذاتي والتكرار Recursion and Iteration

يُستخدم كلٌّ من الاستدعاء الذاتي والتكرار في تنفيذ مجموعة من التعليمات لعدة مرات، والفارق الرئيس بين الاستدعاء الذاتي والتكرار هو طريقة إنهاء الدالة التكرارية. دالة الاستدعاء التكرارية تستدعي نفسها وتُنهي التنفيذ عندما تصل إلى الحالة الأساسية. أما التكرار فيُنفِّذ لبِنَةَ المقطع البرمجي باستمرار حتى يتحقق شرط مُحدَّد أو ينقضي عدد مُحدَّد من التكرارات.

الجدول التالي يعرض بعض الاختلافات بين الاستدعاء الذاتي والتكرار.

## جدول 2.3: التكرار والاستدعاء الذاتي

| التكرار | الاستدعاء الذاتي |
|---|---|
| سريع التنفيذ. | بطيء التنفيذ مقارنةً بالتكرار. |
| يتطلب حجم ذاكرة أقل. | يتطلب حجم ذاكرة أكبر. |
| حجم المقطع البرمجي أكبر. | حجم المقطع البرمجي أصغر. |
| ينتهي باستكمال العدد المُحدَّد من التكرارات أو تحقيق شرط مُعيَّن. | ينتهي بمجرد الوصول إلى الحالة الأساسية. |


وزارة التعليم
Ministry of Education
2025 - 1447

متى تَستخدِم الاستدعاء الذاتي؟

- يُعدُّ الاستدعاء الذاتي الطريقة الأكثر ملائمة للتعامل مع المشكلة في العديد من الحالات.
- يَسهُل استكشاف بعض هياكل البيانات باستخدام الاستدعاء الذاتي.
- بعض **خوارزميات التصنيف** (Sorting Algorithms)، تَستخدِم الاستدعاء الذاتي، مثل: **التصنيف السريع** (Quick Sort).

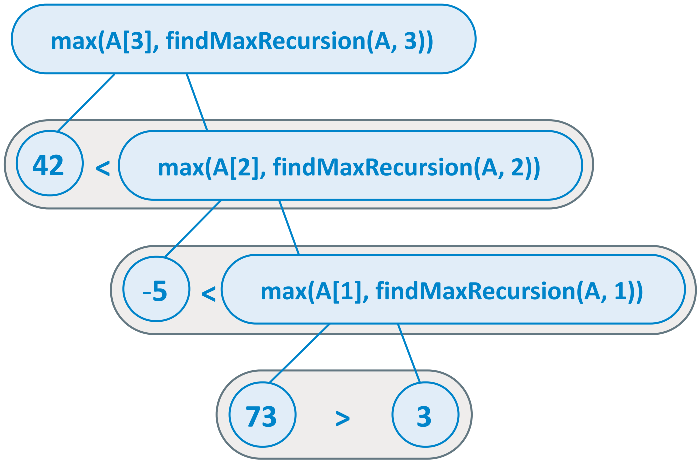
في المثال التالي، ستستخرج أكبر رقم موجود في قائمة مكونة من الأرقام باستخدام دالة الاستدعاء التكرارية.

كما يظهر في السطر الأخير من المثال دالة أخرى للتكرار لغرض المقارنة.

```
def findMaxRecursion(A,n):

    if n==1:
        m = A[n-1]
    else:
        m = max(A[n-1],findMaxRecursion(A,n-1))
    return m

def findMaxIteration(A,n):

    m = A[0]
    for i in range(1,n):
        m = max(m,A[i])
    return m

# main program
myList = [3,73,-5,42]
l = len(myList)
myMaxRecursion = findMaxRecursion(myList,l)
print("Max with recursion is: ", myMaxRecursion)
myMaxIteration = findMaxIteration(myList,l)
print("Max with iteration is: ", myMaxIteration)
```

تستخرج الدالة **max()** العنصر ذا القيمة الأكبر (العنصر ذو القيمة الأكبر في **myList**).

```
Max with recursion is:  73
Max with iteration is:  73
```




شكل 2.5: شجرة الاستدعاء الذاتي لدالة استخراج أكبر رقم في قائمة مكونة من الأرقام

في البرنامج التالي، ستُنشئ دالة استدعاء تكرارية لحساب مُضاعَف الرقم.

ستقوم بإدخال رقمًا (الأساس) وفهرسًا (الأُس أو القُوَّة) يقبلهما البرنامج، ومن ثَمَّ ستَستخدِم دالة الاستدعاء التكرارية ( )powerFunRecursive التي ستَستخدِم هذين المُدخَلَين لحساب مُضاعَف الرقم. يمكن تحقيق الأمر نفسه باستخدام التكرار، والمثال التالي يوضّح ذلك:

```
def powerFunRecursive(baseNum,expNum):
    if(expNum==1):
       return(baseNum)
    else:
       return(baseNum*powerFunRecursive(baseNum,expNum-1))

def powerFunIteration(baseNum,expNum):

    numPower = 1
    for i in range(exp):
        numPower = numPower*base
    return numPower

# main program
base = int(input("Enter number: "))
exp = int(input("Enter exponent: "))
numPowerRecursion = powerFunRecursive(base,exp)
print( "Recursion: ", base, " raised to ", exp, " = ",numPowerRecursion)
numPowerIteration = powerFunIteration(base,exp)
print( "Iteration: ", base, " raised to ", exp, " = ",numPowerIteration)
```

```
Enter number: 10
Enter exponent: 3
Recursion: 10 raised to 3 = 1000
Iteration: 10 raised to 3 = 1000
```

## دالة الاستدعاء التكرارية اللانهائية Infinite Recursive Function

يجب أن تكون حذرًا للغاية عند تنفيذ الاستدعاء التكراري، كما يجب عليك استخدام طريقة معينة لإيقاف التكرار عند تحقيق شرط مُحدَّد لتجنب حدوث الاستدعاء التكراريّ اللانهائيّ، الذي يسبّب توقُّف النظام عن الاستجابة بسبب كثرة استدعاءات الدالة، مما يؤدي إلى **فَيْض الذاكرة** (Memory Overflow) وإنهاء التطبيق.

# تمرينات

1

| حدِّد الجملة الصحيحة والجملة الخاطئة فيما يلي: | صحيحة | خاطئة |
|---|---|---|
| 1. تتكون دالة الاستدعاء التكرارية من حالتين. | ⚪ | ⚪ |
| 2. تستدعي دالة الاستدعاء التكرارية دالة أخرى. | ⚪ | ⚪ |
| 3. دوال الاستدعاء التكرارية أسرع في التنفيذ. | ⚪ | ⚪ |
| 4. استدعاء الدوال يجعل لبِنة المقطع البرمجي أصغر حجمًا. | ⚪ | ⚪ |
| 5. كتابة مقطع برمجي مُتكرِّر يتطلب استدعاءً ذاتيًّا أقل. | ⚪ | ⚪ |

2 ما الاختلافات بين التكرار والاستدعاء الذاتي؟

3 متى يجب استخدام الاستدعاء الذاتي؟



وزارة التعليم
Ministry of Education
2025 - 1447

4 وَضِّح مزايا استخدام الاستدعاء الذاتي وعيوبه.

5 اكتب دالة استدعاء تكرارية بلغة البايثون تقوم بحساب الرقم الأكبر بترتيب محدد (مثلًا ثاني أكبر رقم) في قائمة من الأرقام.

6 اكتُب دالة استدعاء تكرارية بلغة البايثون لحساب مجموع كل الأرقام الزوجية في قائمة معيّنة.

رابط الدرس الرقمي

www.ien.edu.sa

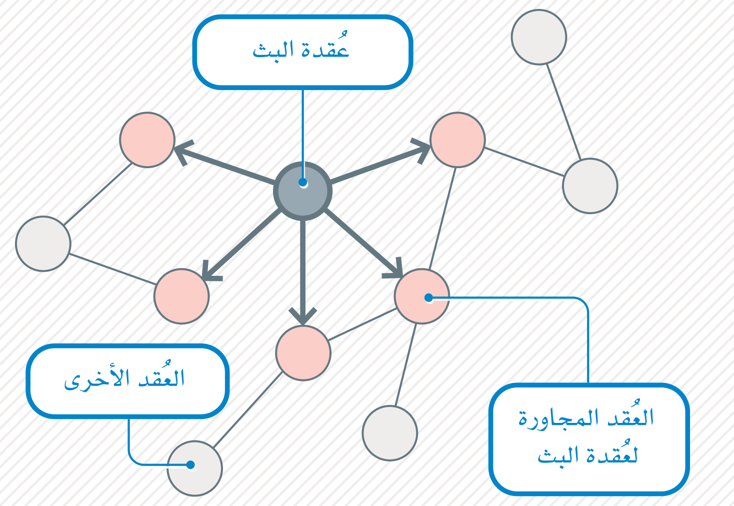
الدرس الثاني

# خوارزمية البحث بأولوية العمق والبحث بأولوية الاتساع

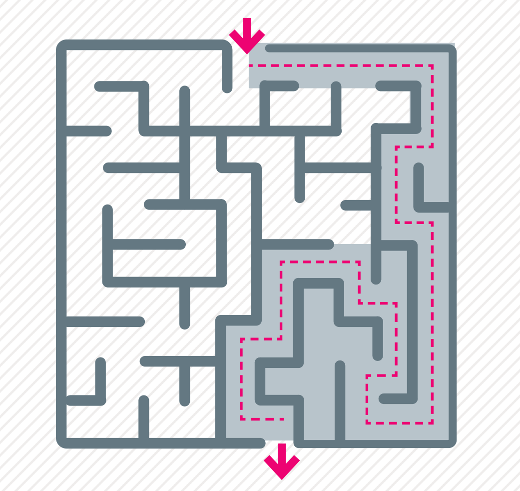
## البحث في المُخطَّطات Searching in Graphs

هناك بعض الحالات التي تحتاج فيها إلى البحث عن عُقدة مُحدَّدة في المُخطَّط، أو تفحُّص كل عُقدة في المُخطَّط لإجراء عملية بعينها مثل طباعة عُقد المُخطَّط، فتكون حالتكَ كشخصٍ يبحث عن المدينة التي يريد السّفر إليها؛ وليتحقق هذا، تحتاج إلى فحص كل عُقدة في المُخطَّط حتى تجد تلك التي تحتاج إليها. يُطلق على هذا الإجراء: البحث في المُخطَّط أو مسح المُخطَّط، وهناك العديد من خوارزميات البحث التي تساعد على تنفيذه، مثل:

- **خوارزمية البحث بأولوية الاتساع** (Breadth-First Search - BFS).
- **خوارزمية البحث بأولوية العمق** (Depth-First Search - DFS).




مثال على خوارزمية البحث بأولوية الاتساع (BFS): البث الشبكي



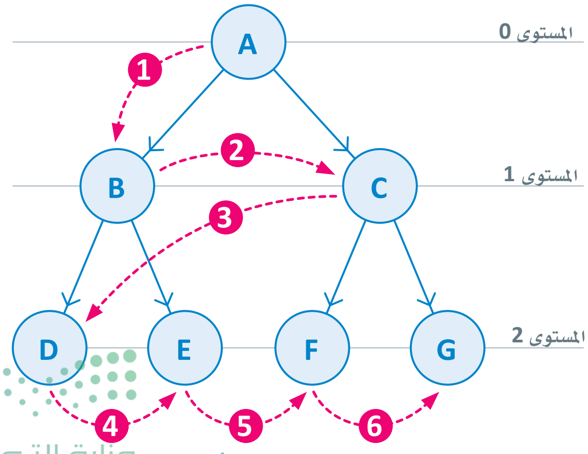
مثال على خوارزمية البحث بأولوية العمق (DFS): حل المتاهة

## خوارزمية البحث بأولوية الاتساع Breadth-First Search (BFS) Algorithm

تستكشف خوارزمية **البحث بأولوية الاتساع** (BFS) المُخطَّط بحسب المستوى واحدًا تلو الآخر، حيث تبدأ بفحص عُقدة الجذر (عُقدة البداية)، ثم تفحص جميع العُقد المرتبطة بها بشكل مباشر واحدة تلو الأخرى.

بعد الانتهاء من فحص كل العُقد في المستوى، تنتقل إلى المستوى التالي، وتتبع الإجراءات نفسها المُوضحّة في الشكل 2.6.

يُستخدَم الطّابور لتتبّع العُقد التي تمّ فحصها، وبمجرّد استكشاف العُقدة، ستتم إضافة العُقَد الفرعية إلى الطابور، ثم تحذف العُقدة التالية الموجودة في أول الطابور التي تم استكشافها سابقًا.

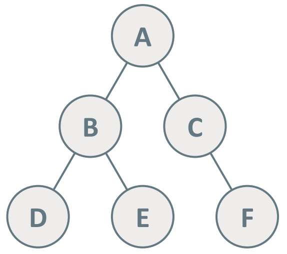



شكل 2.6: خوارزمية البحث بأولوية الاتساع (BFS)

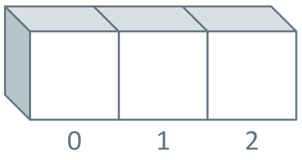

وزارة التعليم
Ministry of Education
2025 - 1447

المثال التالي يوضِّح طريقة عمل خوارزمية **البحث بأولوية الاتساع** (BFS). باستخدام المُخطَّط التالي، حدِّد العُقد التي يجب فحصها للانتقال من عُقدة الجذر A إلى العُقدة F:

ملاحظة: استخدِم هيكل البيانات المُناسب.




المُخطَّط



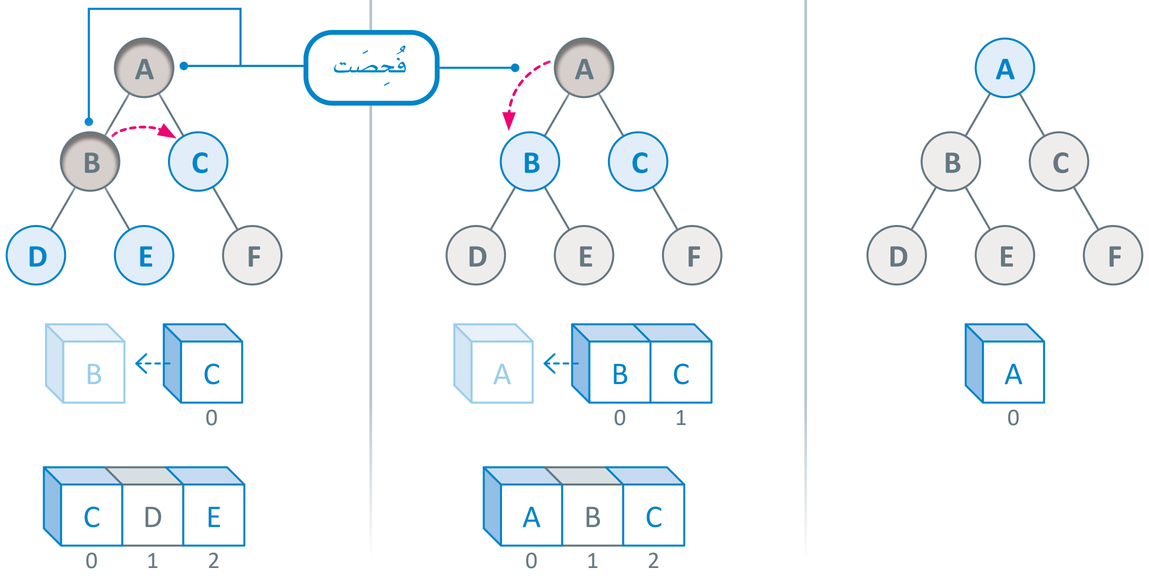

الطابور

**عليك فحص كل العُقد في المستوى 1 قبل الانتقال إلى العُقد في المستوى 2.**

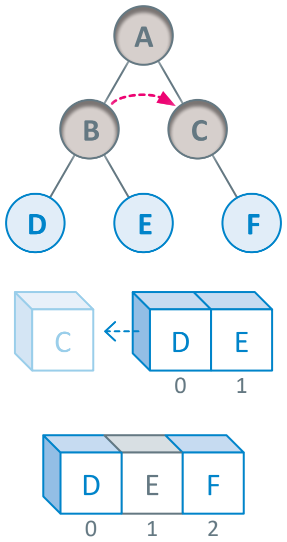
1 البداية من العُقدة الجذرية (العُقدة A). أضف العُقدة الجذريّة إلى الطابور.

2 احذف العُقدة الجذريّة من الطابور لمعالجتها، ثم أضف فروع هذه العُقدة إلى الطابور (العُقدتين B وC).

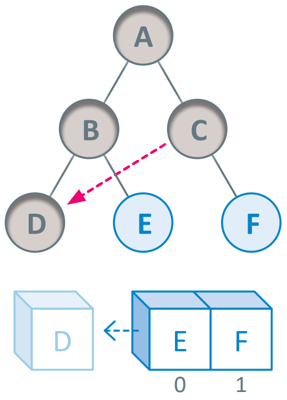
3 احذف العُقدة من مقدمة الطابور (العُقدة B) لمعالجتها، ثم أضف فروع هذه العُقدة إلى الطابور (العُقدتين D وE).



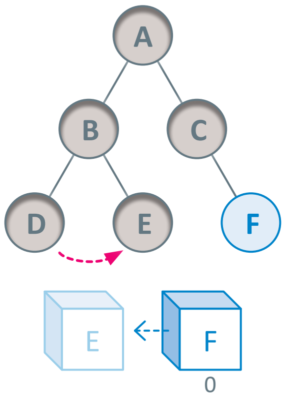
4 احذف العُقدة C وعالجها، ثم أضف فَرعها إليها.



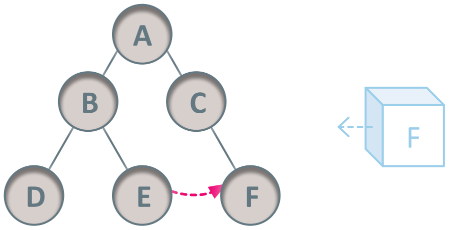

5 احذف العُقدة D لمعالجتها. (ليس لديها فروع).




6 احذف العُقدة E لمعالجتها. (ليس لديها فروع).




7 احذف العُقدة F لمعالجتها، وبذلك أصبح الطابور الآن فارغًا وانتهت عملية البحث.




**العُقد التي فُحصَت باستخدام خوارزمية البحث بأولوية الاتساع (BFS) هي: A، B، C، D، E، F.**

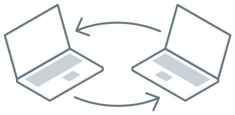
لاحظ كيف يُمكنك تطبيق خوارزمية **البحث بأولوية الاتساع** (BFS) بلغة **البايثون** (Python) في المثال التالي:

```
graph = {
    "A" : ["B","C"],
    "B" : ["D","E"],
    "C" : ["F"],
    "D" : [],
    "E" : [],
    "F" : []
}

visitedBFS = [] # List to keep track of visited nodes
queue = []      # Initialize a queue

 # bfs function
def bfs(visited, graph, node):
    visited.append(node)
```

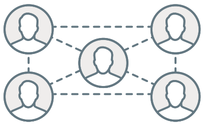
```
  queue.append(node)

  while queue:
    n = queue.pop(0)
    print (n, end = " ")

    for neighbor in graph[n]:
      if neighbor not in visited:
        visited.append(neighbor)
        queue.append(neighbor)

# main program
bfs(visitedBFS, graph, "A")
```

```
A B C D E F
```

## التطبيقات العملية لخوارزمية البحث بأولوية الاتساع
## Practical Applications of the BFS Algorithm

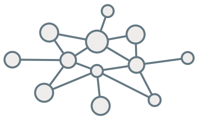


تُستخدَم في **شبكات النّظير للنّظير** (Peer-to-Peer Networks) للعثور على كل العُقد المجاورة من أجل تأسيس الاتصال.



تُستخدَم في **وسائل التواصل الاجتماعي** (Social Media) لربط عُقد المُستخدِمين المُرتبطين، مثل أولئك الذين لهم الاهتمامات نفسها أو الموقع نفسه.

تُستخدَم في **نُظم الملاحة باستخدام مُحدِّد المواقع العالمي** (GPS Navigation Systems) للبحث عن الأماكن المتجاورة حتى تُحدِّد الاتجاهات التي يتبعها المُستخدِم.



تُستخدَم للحصول على **البث الشبكي** (Network Broadcasting) لبعض الحُزم.

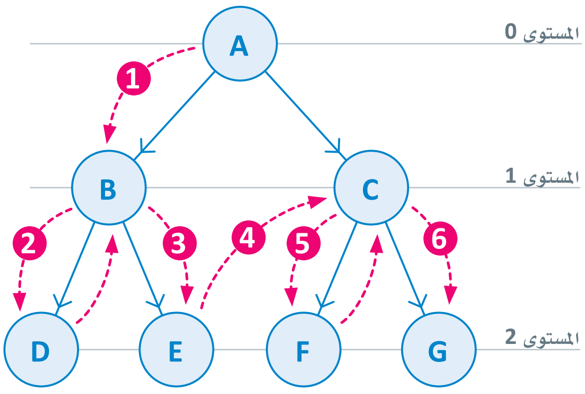
**معلومة**

يُمكن تطوير خوارزمية البحث بأولوية الاتساع (BFS) بتحديد نقطة البداية (الحالة الأوّليّة) ونقطة الهدف (الحالة المُستهدَفة) لإيجاد المسار بينهما.

وزارة التعليم
Ministry of Education
2025 - 1447

## خوارزمية البحث بأولوية العمق
## Depth-First Search (DFS) Algorithm

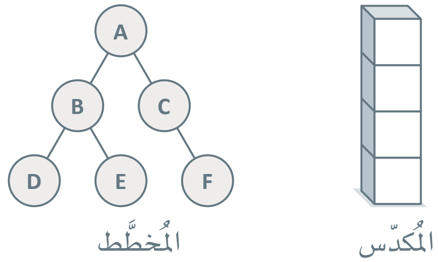
في **البحث بأولوية العمق** (DFS)، ستقوم باتباع الحواف، وتتعمق أكثر وأكثر في المُخطَّط. يَستخدِم البحث بأولوية العمق إجراء استدعاء تكراري للتنقل عبر العُقد. عند الوصول إلى عُقدة لا تحتوي على حواف لأي عُقدة جديدة، ستعود إلى العُقدة السابقة وتستمر العملية. تَستخدِم خوارزمية البحث بأولوية العمق هيكل بيانات المُكدّس لتتبع مسار الاستكشاف. بمجرد استكشاف عُقدة، ستُضاف إلى المُكدّس. عندما ترغب في العودة، ستحذف العُقدة من المُكدّس كما هو موضَّح في الشكل 2.7.



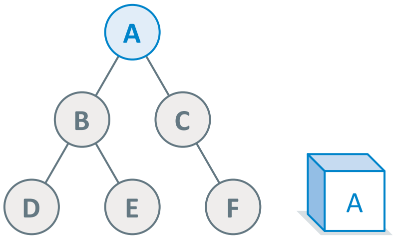
شكل 2.7: خوارزمية البحث بأولوية العمق (DFS)

المثال التالي يوضِّح طريقة عمل خوارزمية **البحث بأولوية العمق** (DFS)، باستخدام المُخطَّط التالي، تَتَبّع ترتيب استكشاف العُقد (Traversal) بحسب خوارزمية البحث بأولوية العمق.

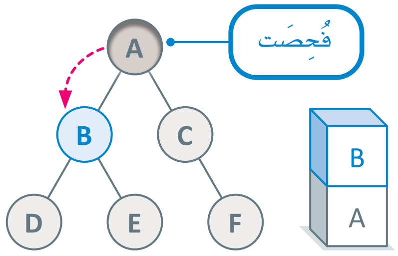
ملاحظة: استخدِم هيكل البيانات المُناسب.



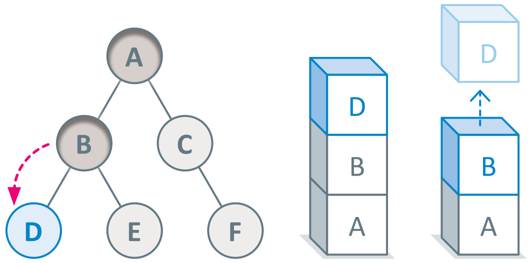

**1** عالِج الجذر A ثم أضفه إلى المُكدّس.



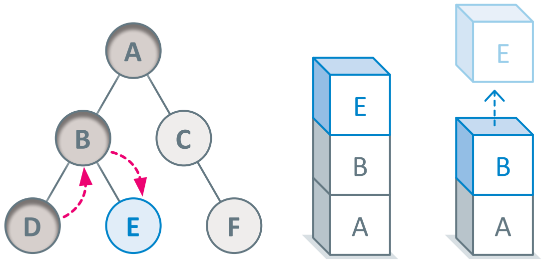

**2** عالِج العُقدة B ثم أضفها إلى المُكدّس.




**3** عالِج العُقدة D ثم أضفها إلى المُكدّس. ستُحذَف العُقدة التي فُحِصَت وليس لها فروع من المُكدّس. (احذف العُقدة D).




**4** عالِج العُقدة E ثم أضفها إلى المُكدّس. ستُحذَف العُقدة التي فُحِصَت وليس لها فروع من المُكدّس. (احذف العُقدة E).

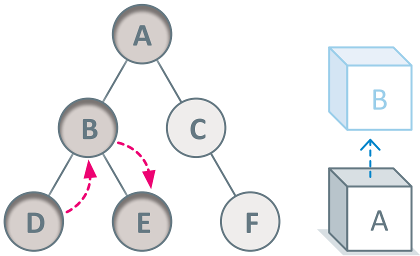



### لمحة تاريخية

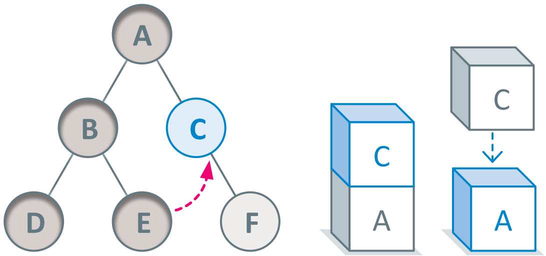
طُوِّرَت النسخة الأولى من خوارزمية البحث بأولوية العمق (DFS) في القرن التاسع عشر بواسطة عالم رياضيات فرنسي كاستراتيجية لحل المتاهات.


وزارة التعليم
Ministry of Education
2025 - 1447

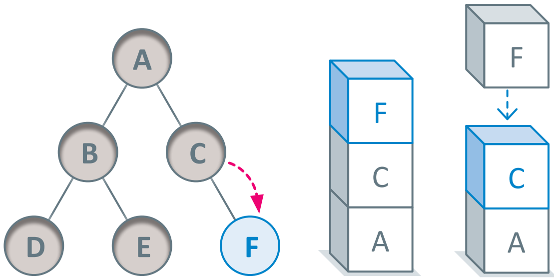
5 احذف العُقدة B.



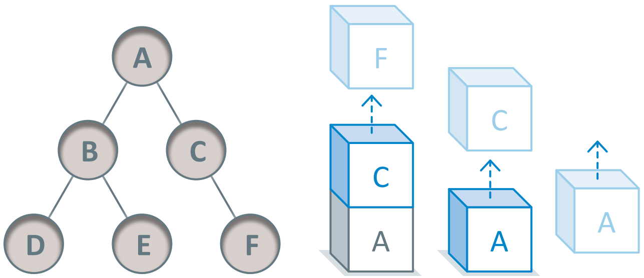

6 عالِج العُقدة C ثم أضفها إلى المُكدّس.




7 عالِج العُقدة F ثم أضفها إلى المُكدّس.




8 المُكدّس خالي وبالتالي ستتوقف خوارزمية **البحث بأولوية العمق** (DFS).




والآن ستتعلّم طريقة تنفيذ خوارزمية **البحث بأولوية العمق** (DFS) في لغة البايثون.

**العُقد التي فُحِصَت باستخدام خوارزمية البحث بأولوية العمق (DFS) هي: F ،C ،E ،D ،B ،A.**

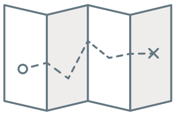
```
graph = {
    "A" : ["B","C"],
    "B" : ["D","E"],
    "C" : ["F"],
    "D" : [],
    "E" : [],
    "F" : []
}

visitedDFS = [] # list to keep track of visited nodes

# dfs function
def dfs(visited, graph, node):
    if node not in visited:
        print(node, end = " ")
        visited.append(node)
        for neighbor in graph[node]:
            dfs(visited, graph, neighbor)


# main program
dfs(visitedDFS, graph, "A")
```

يُستخدم المُكدّس بصورة غير مباشرة عبر **مُكدّس أثناء التشغيل** (Runtime Stack) لتتبُّع الاستدعاءات التكرارية.

```
A B D E C F
```

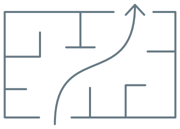
وزارة التعليم
Ministry of Education
2025 - 1447

## التطبيقات العملية لخوارزمية البحث بأولوية العمق
## Practical Applications of the DFS Algorithm



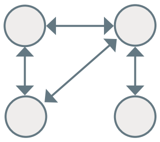
تُستخدَم خوارزمية البحث بأولوية العمق في **إيجاد المسارات** (Path Finding) لاستكشاف المسارات المختلفة في العمق للخرائط والطرقات والبحث عن المسار الأفضل.

---



تُستخدَم خوارزمية البحث بأولوية العمق في **حل المتاهات** (Solve Mazes) من خلال اجتياز كل الطُرُق الممكنة.

---



يُمكِن تحديد **الدورات** (Cycles) في المُخطَّط باستخدام خوارزمية البحث بأولوية العمق من خلال وجود **حافة خلفية** (Back Edge)، تمُر من خلال العُقدة نفسها مرتين.

**جدول 2.4: مقارنة بين خوارزمية البحث بأولوية الاتساع (BFS) وخوارزمية البحث بأولوية العمق (DFS)**

| معايير المقارنة | خوارزمية البحث بأولوية الاتساع (BFS) | خوارزمية البحث بأولوية العمق (DFS) |
| --- | --- | --- |
| طريقة التنفيذ | التنقّل حسب مستوى الشجرة. | التنقّل حسب عُمق الشجرة. |
| هيكل البيانات | تَستخدِم هيكل بيانات الطابور لتتبُّع الموقع التالي لفحصه. | تَستخدِم هيكل بيانات المُكدّس لتتبُّع الموقع التالي لفحصه. |
| الاستخدام | يُفضَّل استخدامها عندما يكون هيكل المُخطَّط واسعًا وقصيرًا. | يُفضَّل استخدامها عندما يكون هيكل المُخطَّط ضيقًا وطويلًا. |
| طريقة البحث | تبحث عن مسار الوجهة باستخدام أقل عدد من الحواف. | يتجة البحث إلى قاع الشجرة الفرعية، ثم يتراجع. |
| العُقد التي تُفحص في البداية | فحص عُقد الأشقاء قبل الفروع. | فحص عُقد الفروع قبل الأشقاء. |


وزارة التعليم
Ministry of Education
2025 - 1447

# تمرينات

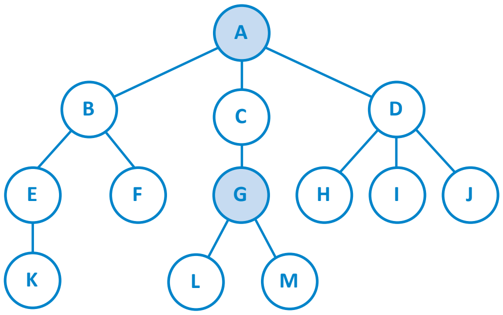
1

| حدِّد الجملة الصحيحة والجملة الخاطئة فيما يلي: | صحيحة | خاطئة |
|---|---|---|
| 1. تُنفَّذ خوارزمية البحث بأولوية الاتساع (BFS) وخوارزمية البحث بأولوية العمق (DFS) باستخدام الاستدعاء الذاتي. | ○ | ○ |
| 2. لا يمكن استخدام خوارزمية البحث بأولوية الاتساع (BFS) وخوارزمية البحث بأولوية العمق (DFS) في هيكل بيانات الشجرة. | ○ | ○ |
| 3. تُنفَّذ خوارزمية البحث بأولوية الاتساع (BFS) بمساعدة هيكل بيانات القائمة المترابطة. | ○ | ○ |
| 4. يمكن تنفيذ خوارزمية البحث بأولوية العمق (DFS) بمساعدة هيكل بيانات المُكدّس. | ○ | ○ |
| 5. لا يمكن استخدام خوارزمية البحث بأولوية الاتساع (BFS) في البث الشبكي. | ○ | ○ |

2 اشرح كيف تعمل خوارزمية البحث بأولوية الاتساع (BFS) وخوارزمية البحث بأولوية العمق (DFS).

3 قارن بين خوارزمية البحث بأولوية الاتساع (BFS) وخوارزمية البحث بأولوية العمق (DFS).

**4** في المُخطَّط على اليسار، انتقِل من عُقدة البداية A إلى عُقدة الهدف G. طبِّق خوارزمية البحث بأولوية الاتساع (BFS) وخوارزمية البحث بأولوية العمق (DFS) باستخدام هيكل البيانات المناسب (المُكدّس أو الطابور)، مع الإشارة إلى العُقد التي فُحِصَت.



5 اكتب دالة بلغة البايثون تَستخدِم خوارزمية البحث بأولوية الاتساع (BFS) في مُخطَّط للتحقق مما إذا كان هناك مسارٌ بين عُقدتين مُعطاتين.

6 اكتب دالة بلغة البايثون تَستخدِم خوارزمية البحث بأولوية العمق (DFS) لإيجاد المسار الأقصر في مُخطَّط غير موزون.

وزارة التعليم
Ministry of Education
2025 - 1447

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الثالث

# اتخاذ القرار القائم على القواعد

## الأنظمة القائمة على القواعد Rule-Based Systems

تُركِّز أنظمة الذكاء الاصطناعي القائمة على القواعد على استخدام مجموعة من القواعد المُحدَّدة مُسبقًا لاتخاذ القرارات وحل المشكلات. **الأنظمة الخبيرة** (Expert Systems) هي المثال الأكثر شهرة للذكاء الاصطناعي القائم على القواعد، وهي إحدى صور الذكاء الاصطناعي الأولى التي طُوِّرت وانتشرت في فترة الثمانينيات والتسعينيات من القرن الماضي. وغالبًا ما كانت تُستخدَم لأتمتة المهام التي تتطلب عادةً خبرات بشرية مثل: تشخيص الحالات الطبية أو تحديد المشكلات التقنية وإصلاحها. واليوم لم تَعُد الأنظمة القائمة على القواعد التقنية هي الأحدث، حيث تفوّقت عليها منهجيات الذكاء الاصطناعي الحديثة. ومع ذلك، لا تزال الأنظمة الخبيرة شائعة الاستخدام في العديد من المجالات نظرًا لقدرتها على الجمع بين الأداء المعقول وعملية اتخاذ القرار البديهية والقابلة للتفسير.

### قاعدة المعرفة Knowledge Base

أحد المكونات الرئيسة لأنظمة الذكاء الاصطناعي القائمة على القواعد هي قاعدة المعرِفة، وهي مجموعة من الحقائق والقواعد التي يَستخدِمها النظام لاتخاذ القرارات. تُدخَل هذه الحقائق والقواعد في النظام بواسطة الخبراء البشريين المسؤولين عن تحديد المعلومات الأكثر أهمية وتحديد القواعد التي يتَّبعها النظام. لاتخاذ القرار أو حل المُشكلة، يبدأ النظام الخبير بالتحقق من الحقائق والقواعد في قاعدة البيانات وتطبيقها على الموقف الحالي. إن لم يتمكن النظام من العثور على تطابق بين الحقائق والقواعد في قاعدة المعرِفة، فقد يطلب من المُستخدِم معلومات إضافية أو إحالة المشكلة إلى خبير بشري لمزيد من المساعدة، وإليكَ بعض مزايا وعيوب الأنظمة القائمة على القواعد موضحة في الجدول 2.5:

**الأنظمة الخبيرة (Expert Systems):**

النظام الخبير هو أحد أنواع الذكاء الاصطناعي الذي يُحاكي قدرة اتخاذ القرار لدى الخبير البشري. يَستخدِم النظام قاعدة المعرفة المُكوَّنة من قواعد وحقائق ومحركات الاستدلال لتقديم المشورة أو حل المشكلات في مجال معرفي مُحدَّد.

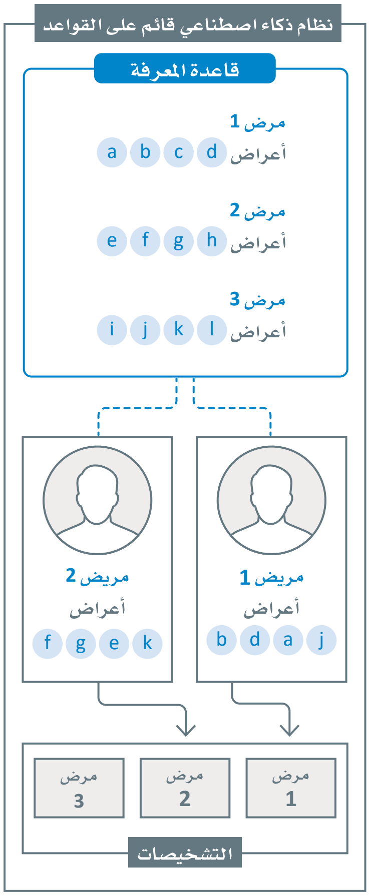
**جدول 2.5: المزايا والعيوب الرئيسة للأنظمة القائمة على القواعد**

| المزايا | العيوب |
| --- | --- |
| • يُمكنها اتخاذ القرارات وحل المشكلات بسرعة وبدقة أفضل من البشر، خاصةً عندما يتعلق الأمر بالمهام التي تتطلب قدرًا كبيرًا من المعرِفة أو البيانات.<br>• تَعمل هذه الأنظمة باستمرار، دون تحيُّز أو أخطاء قد تؤثر في بعض الأحيان على اتخاذ القرار البشري. | • تَعمل هذه الأنظمة بكفاءة طالما كانت مُدخَلات المعرِفة والقواعد جيدة، وقد لا تستطيع التعامل مع المواقف التي تقع خارج نطاق خبراتها.<br>• لا يُمكنها التعلُّم أو التكيُّف بالطريقة نفسها مثل البشر، وهذا يجعلها أقل قابلية للتطبيق على الأحداث المُتغيِّرة حيث تتغير مُدخَلات البيانات والمنطق كثيرًا بمرور الوقت. |



وزارة التعليم
Ministry of Education
2025 - 1447

في هذا الدرس ستتعلّم المزيد حول الأنظمة القائمة على القواعد في سياق أحد تطبيقاتها الرئيسة، وهو: التشخيص الطبي. سيعرض النظام تشخيصًا طبيًا وفقًا للأعراض التي تظهر على المريض، كما هو مُوضّح في الشكل 2.8. بدءًا بنظام تشخيص بسيط مُستنِد إلى القواعد، وستكتشف بعض الأنظمة الأكثر ذكاءً وكيف يُحقِّق كل تكرار نتائج أفضل.




شكل 2.8: التشخيص الطبي بواسطة نظام الذكاء الاصطناعي القائم على القواعد

## الإصدار 1

في الإصدار الأول ستبني نظامًا بسيطًا قائمًا على القواعد يمكنه تشخيص ثلاثة أمراض مُحتملة: KidneyStones (حصى الكُلى)، وAppendicitis (التهاب الزائدة الدودية)، وFood Poisoning (التسمُّم الغذائي). ستكون المُدخَلات إلى النظام هي قاعدة معرفة بسيطة تربط كل مرض بقائمة من الأعراض المُحتملة. يتوفّر ذلك في ملف بتنسيق JSON (جيسون) يُمكنك تحميله وعرضه كما هو مُوضَّح بالأسفل.

```python
import json  # a library used to save and load JSON files

# the file with the symptom mapping
symptom_mapping_file='symptom_mapping_v1.json'

# open the mapping JSON file and load it into a dictionary
with open(symptom_mapping_file) as f:
    mapping=json.load(f)

# print the JSON file
print(json.dumps(mapping, indent=2))
```

```
{
  "diseases": {
    "food poisoning": [
      "vomiting",
      "abdominal pain",
      "diarrhea",
      "fever"
    ],
    "kidney stones": [
      "lower back pain",
      "vomiting",
      "fever"
    ],
    "appendicitis": [
      "abdominal pain",
      "vomiting",
      "fever"
    ]
  }
}
```

وزارة التعليم
Ministry of Education
2025 - 1447

سيتَّبع الإصدار الأول القائم على القواعد قاعدة بسيطة ألا وهي: إذا كان لدى المريض على الأقل ثلاثًا من جميع الأعراض المحتملة للمرض، فيجب إضافة المرض كتشخيص مُحتَمل. يمكنك العثور أدناه على دالة Python (البايثون) التي تَستخدِم هذه القاعدة لإجراء التشخيص، بالاستناد إلى قاعدة المعرِفة المذكورة أعلاه وأعراض المرض الظاهرة على المريض.

```
def diagnose_v1(patient_symptoms:list):

    diagnosis=[] # the list of possible diseases

    if "vomiting" in patient_symptoms:

        if "abdominal pain" in patient_symptoms:

            if "diarrhea" in patient_symptoms:

                # 1:vomiting, 2:abdominal pain, 3:diarrhea
                diagnosis.append('food poisoning')

            elif 'fever' in patient_symptoms:

                # 1:vomiting, 2:abdominal pain, 3:fever
                diagnosis.append('food poisoning')
                diagnosis.append('appendicitis')

        elif "lower back pain" in patient_symptoms and 'fever' in patient_symptoms:

            # 1:vomiting, 2:lower back pain, 3:fever
            diagnosis.append('kidney stones')

    elif "abdominal pain" in  patient_symptoms and\
         "diarrhea" in patient_symptoms and\
         "fever" in patient_symptoms:\
        # 1:abdominal pain, 2:diarrhea, 3:fever
        diagnosis.append('food poisoning')

    return diagnosis
```

في هذه الحالة، تكون قاعدة المعرِفة محددةً **بتعليمات برمجية ثابتة** (Hard-Coded) داخل الدالة في شكل عبارات IF. تَستخدِم هذه العبارات الأعراض الشائعة بينَ الأمراضِ الثلاثة للتوصل تدريجيًا إلى التشخيص في أسرع وقت ممكن. على سبيل المثال، عَرض Vomiting (القيء) مشترك بين جميع الأمراض. لذلك، إذا كانت عبارة IF الأولى صحيحة فقد تم بالفعل حساب أحد الأعراض الثلاثة المطلوبة لجميع الأمراض. بعد ذلك، ستبدأ في البحث عن Abdominal Pain (ألم البطن) المرتبط بمرضين وتستمر بالطريقة نفسها حتى يتم النظر في جميع مجموعات الأعراض الممكنة.


وزارة التعليم
Ministry of Education
2025 - 1447

يُمكنك بعد ذلك اختبار هذه الدالة على ثلاثة مرضى مختلفين:

```
# Patient 1
my_symptoms=['abdominal pain', 'fever', 'vomiting']
diagnosis=diagnose_v1(my_symptoms)
print('Most likely diagnosis:',diagnosis)

# Patient 2
my_symptoms=['vomiting', 'lower back pain', 'fever' ]
diagnosis=diagnose_v1(my_symptoms)
print('Most likely diagnosis:',diagnosis)

# Patient 3
my_symptoms=['fever', 'cough', 'vomiting']
diagnosis=diagnose_v1(my_symptoms)
print('Most likely diagnosis:',diagnosis)
```

```
Most likely diagnosis: ['food poisoning', 'appendicitis']
Most likely diagnosis: ['kidney stones']
Most likely diagnosis: []
```

شكل 2.9: تمثيل الإصدار الأول

يتضمن التشخيص الطبي للمريض الأول التسمُّم الغذائي والتهاب الزائدة الدودية؛ لأن الأعراض الثلاثة التي تظهر على المريض ترتبط بكلا المرضين. يُشخَّص المريض الثاني بحصى الكُلى، فهو المرض الوحيد الذي تجتمع فيه الأعراض الثلاثة. في النهاية، لا يُمكن تشخيص الحالة الطبية للمريض الثالث؛ لأن الأعراض الثلاثة التي ظهرت على المريض لا تجتمع في أيٍ من الأمراض الثلاثة.

يتميز الإصدار الأول القائم على القواعد بالبديهية والقابلية للتفسير، كما يتضمن استخدام قاعدة المعرفة والقواعد في التشخيص الطبي دون تَحيُّز أو انحراف عن الخط المعياري. ومع ذلك، يشوب هذا الإصدار العديد من العيوب: أولًا، أن قاعدة ثلاثة أعراض على الأقل هي تمثيل مُبسَّط للغاية لكيفية التشخيص الطبي على يد الخبير البشري. ثانيًا، أن قاعدة المعرفة داخل الدالة تكون محددةً بتعليماتٍ برمجيةٍ ثابتة، وعلى الرغم من أنه يسهُل إنشاء عبارات شَرطيَّة بسيطة لقواعد المعرفة الصغيرة، إلا أن المهمة تصبح أكثر تعقيدًا وتستغرق وقتًا طويلًا عند تشخيص الحالات التي تعاني من العديد من الأمراض والأعراض المرضية.


وزارة التعليم
Ministry of Education
2025 - 1447

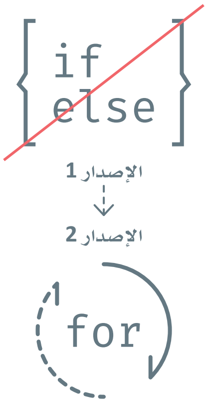
## الإصدار 2

في الإصدار الثاني، ستُعزِّز مرونة وقابلية تطبيق النظام القائم على القواعد بتمكينه من قراءة قاعدة المعرِفة المُتغيِّرة مباشرةً من ملف JSON (جسون). سيؤدي هذا إلى الحد من عملية الهندسة اليدوية لعبارات IF الشَّرطيَّة حسب الأعراض ضمن الدالة. وهذا يُعدُّ تحسُّنًا كبيرًا يجعل النظام قابلًا للتطبيق على قواعد المعرِفة الأكبر حجمًا مع تزايد عدد الأمراض والأعراض. وفي الأسفل، مثال يوضِّح قاعدة المعرِفة.

```python
symptom_mapping_file='symptom_mapping_v2.json'

with open(symptom_mapping_file) as f:
    mapping=json.load(f)

print(json.dumps(mapping, indent=2))
```

```
{
  "diseases": {
    "covid19": [
      "fever",
      "headache",
      "tiredness",
      "sore throat",
      "cough"
    ],
    "common cold": [
      "stuffy nose",
      "runny nose",
      "sneezing",
      "sore throat",
      "cough"
    ],
    "flu": [
      "fever",
      "headache",
      "tiredness",
      "stuffy nose",
      "sneezing",
      "sore throat",
      "cough",
      "runny nose"
    ],
    "allergies": [
      "headache",
      "tiredness",
      "stuffy nose",
      "sneezing",
      "cough",
      "runny nose"
    ]
  }
}
```

قاعدة المعرِفة الجديدة هذه أكبر قليلًا من سابقتها. ومع ذلك، يتَّضح أن محاولة إنشاء عبارات IF الشَّرطيَّة في هذه الحالة ستكون أصعب بكثير. على سبيل المثال، تضمنت قاعدة المعرِفة السابقة ربط أحد الأمراض بأربعة أعراض، ومرضين بثلاثة أعراض. وعند تطبيق قاعدة ثلاثة أعراض على الأقل المُطبَّقة في الإصدار الأول، تحصل على 6 مجموعات ثلاثية من الأعراض المحتملة التي تؤخَذ في الاعتبار. في قاعدة المعرِفة الجديدة بالأعلى، تكون للأمراض الأربعة 5 و5 و8 و6 أعراض، على التوالي. وبهذا، تحصل على 96 مجموعة ثلاثية من الأعراض المحتملة. وفي حال التعامل مع مئات أو حتى آلاف الأمراض، ستجدُ أنّه من المستحيل إنشاء نظام مثل الموجود في الإصدار الأول.



شكل 2.10: الإصدار الثاني لا يحتوي على عبارات IF الشَّرطيَّة المحددة بتعليمات برمجية ثابتة

وكذلك، لا يوجد سبب طبي وجيه لِقِصَر التشخيص الطبي على مجموعات ثلاثية من الأعراض. ولذلك، ستجعل **منطق التشخيص** (Diagnosis Logic) أكثر تنوعًا بحساب عدد الأعراض المُطابقة لكل مرض، والسماح للمُستخدِم بتحديد عدد الأعراض المُطابقة التي يجب توافرها في المرض لتضمينه في التشخيص.


وزارة التعليم
Ministry of Education
2025 - 1447

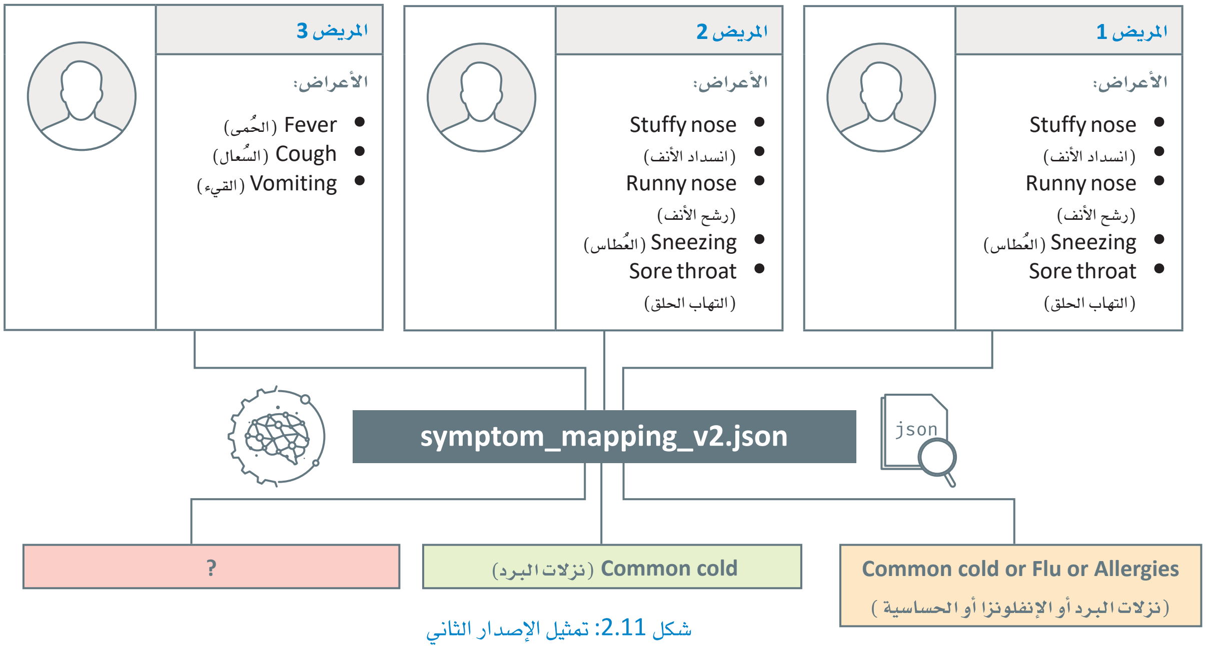
```python
def diagnose_v2(patient_symptoms:list,
                symptom_mapping_file:str,
                matching_symptoms_lower_bound:int):

    diagnosis=[]

    with open(symptom_mapping_file) as f:
        mapping=json.load(f)

    # access the disease information
    disease_info=mapping['diseases']

    # for every disease
    for disease in disease_info:

        counter=0

        disease_symptoms=disease_info[disease]

        # for each patient symptom
        for symptom in patient_symptoms:

            # if this symptom is included in the known symptoms for the disease
            if symptom in disease_symptoms:

                counter+=1

        if counter>=matching_symptoms_lower_bound:
            diagnosis.append(disease)

    return diagnosis
```

لا يحتوي هذا الإصدار على عبارات IF الشَرطيَّة المحددة بتعليماتٍ برمجيةٍ ثابتة. بعد تحميل مُخطَّط الأعراض من ملف JSON (جسون)، يبدأ الإصدار في أخذ كلّ مرض محتمل في الاعتبار باستخدام حلقة التكرار الأولى FOR. تتحقق الحلقة من كل عَرَض على حدة بمقارنته بالأعراض المعروفة للمرض وزيادة **العدَّاد** (Counter) في كل مرة يجد فيها النظام تطابقًا.

وزارة التعليم
Ministry of Education
2025 - 1447

```python
# Patient 1
my_symptoms=["stuffy nose", "runny nose", "sneezing", "sore throat"]
diagnosis=diagnose_v2(my_symptoms,'symptom_mapping_v2.json' , 3)
print('Most likely diagnosis:',diagnosis)

# Patient 2
my_symptoms=["stuffy nose", "runny nose", "sneezing", "sore throat"]
diagnosis=diagnose_v2(my_symptoms, 'symptom_mapping_v2.json' , 4)
print('Most likely diagnosis:',diagnosis)

# Patient 3
my_symptoms=['fever', 'cough', 'vomiting']
diagnosis=diagnose_v2(my_symptoms, 'symptom_mapping_v2.json' , 3)
print('Most likely diagnosis:',diagnosis)
```

```
Most likely diagnosis: ['common cold', 'flu', 'allergies']
Most likely diagnosis: ['common cold']
Most likely diagnosis: []
```




شكل 2.11: تمثيل الإصدار الثاني

لاحظ أن الإصدار الثاني هو نسخة مُعمَّمة من الإصدار الأول. ومع ذلك، يُعدُّ هذا الإصدار أكثر قابلية للتطبيق على نطاق واسع، ويمكن استخدامه كما هو مع أي قاعدة معرفة أخرى بالتنسيق نفسه، حتى لو كانت تشمل الآلاف من الأمراض مع عدد ضخم من الأعراض. كما يَسمح للمُستخدِم بزيادة أو تقليل عدد القيود على التشخيص بضبط المُتغيِّر matching_symptoms_lower_bound. يمكن ملاحظة ذلك في حالة المريض 1 والمريض 2: فعلى الرغم من أنهما يعانيان من الأعراض نفسها، إلا أنه عند ضبط هذا المُتغيِّر، ستحصل على تشخيص مختلف تمامًا.

على الرغم من هذه التحسينات،إلا إنّ بعض العيوب لا تزال موجودة في هذا الإصدار، ولا يُعدُّ تمثيلًا دقيقًا للتشخيص الطبي الحقيقي.

## الإصدار 3

في الإصدار الثالث، ستزيد من ذكاء النظام القائم على القواعد بمنحه إمكانية الوصول إلى نوع مُفصَّل من قاعدة المعرِفة. هذا النوع الجديد يأخذ بعين الاعتبار الحقيقة الطبية التي تقول: إنّ بعض الأعراض تكون أكثر شيوعًا من أخرى للمرض نفسه.

```python
symptom_mapping_file='symptom_mapping_v3.json'

with open(symptom_mapping_file) as f:
    mapping=json.load(f)

print(json.dumps(mapping, indent=2))
```

```
{
  "diseases": {
    "covid19": {
      "very common": [
        "fever",
        "tiredness",
        "cough"
      ],
      "less common": [
        "headache",
        "sore throat"
      ]
    },
    "common cold": {
      "very common": [
        "stuffy nose",
        "runny nose",
        "sneezing",
        "sore throat"
      ],
      "less common": [
        "cough"
      ]
    },
   "flu": {
      "very common": [
        "fever",
        "headache",
        "tiredness",
        "sore throat",
        "cough"
      ],
      "less common": [
        "stuffy nose",
        "sneezing",
        "runny nose"
      ]
    },
    "allergies": {
      "very common": [
        "stuffy nose",
        "sneezing",
        "runny nose"
      ],
      "less common": [
        "headache",
        "tiredness",
        "cough"
      ]
    }
  }
}
```

لن يُنظر إلى المنطق الذي يقتصر على عدد الأعراض، وسيُستبدَل بدالة تسجيل النقاط التي تعطي أوزانًا مُخصَّصة للأعراض الأكثر والأقل شيوعًا. ستَتوفر للمُستخدِم كذلك المرونة لتحديد الأوزان التي يراها مناسبة. سيتمّ تضمين المرض أو الأمراض ذات المجموع الموزون الأعلى في التشخيص.

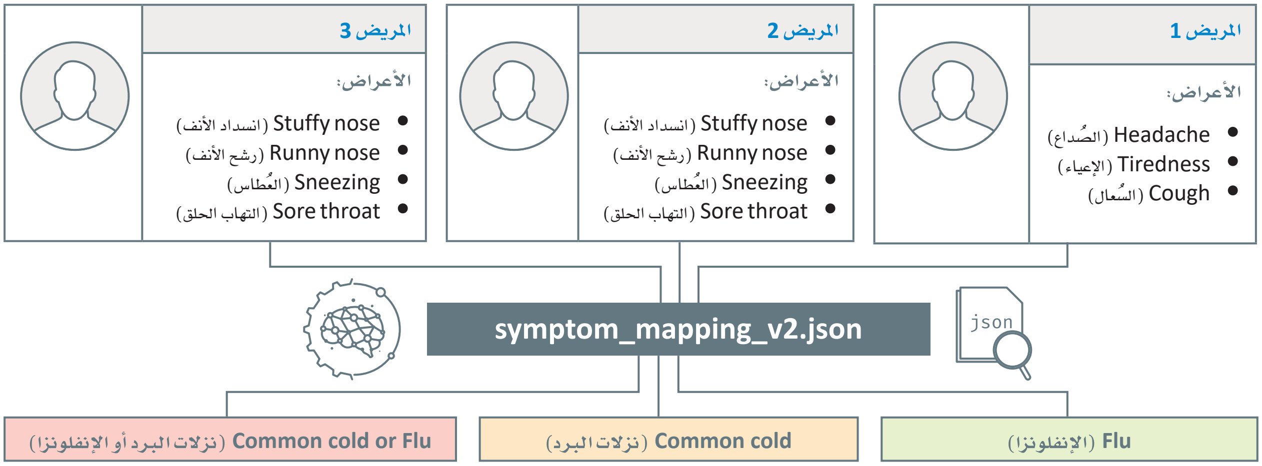
```
from collections import defaultdict

def diagnose_v3(patient_symptoms:list,
                symptom_mapping_file:str,
                very_common_weight:float=1,
                less_common_weight:float=0.5
               ):

    with open(symptom_mapping_file) as f:
        mapping=json.load(f)

    disease_info=mapping['diseases']

    # holds a symptom-based score for  each potential disease
    disease_scores=defaultdict(int)

    for disease in disease_info:

        # get the very common symptoms of the disease
        very_common_symptoms=disease_info[disease]['very common']

        # get the less common symptoms for this disease
        less_common_symptoms=disease_info[disease]['less common']

        for symptom in patient_symptoms:

            if symptom in very_common_symptoms:
                disease_scores[disease]+=very_common_weight

            elif symptom in less_common_symptoms:
                disease_scores[disease]+=less_common_weight

    # find the max score all candidate diseases
    max_score=max(disease_scores.values())

    if max_score==0:
        return []

    else:
        # get all diseases that have the max score
        diagnosis=[disease for disease in disease_scores if disease_scores
[disease]==max_score]

        return diagnosis, max_score
```


وزارة التعليم
Ministry of Education
2025 - 1447

لكل مرض محتمل في قاعدة المعرِفة، تُحدِّد هذه الدالة الجديدة الأعراض الأكثر والأقل ظهورًا على المريض، ثم تزيد من درجة المرض وفقًا لـلأوزان المُقابِلة، وفي الأخير تُدرَج الأمراض ذات الدرجة الأعلى في التشخيص. يُمكنك الآن اختبار تنفيذ الدالة مع بعض الأمثلة:

```
# Patient 1
my_symptoms=["headache", "tiredness", "cough"]
diagnosis=diagnose_v3(my_symptoms, 'symptom_mapping_v3.json')
print('Most likely diagnosis:',diagnosis)

# Patient 2
my_symptoms=["stuffy nose", "runny nose", "sneezing", "sore throat"]
diagnosis=diagnose_v3(my_symptoms, 'symptom_mapping_v3.json')
print('Most likely diagnosis:',diagnosis)

# Patient 3
my_symptoms=["stuffy nose", "runny nose", "sneezing", "sore throat"]
diagnosis=diagnose_v3(my_symptoms, 'symptom_mapping_v3.json', 1, 1)
print('Most likely diagnosis:',diagnosis)
```

```
Most likely diagnosis: (['flu'], 3)
Most likely diagnosis: (['common cold'], 4)
Most likely diagnosis: (['common cold', 'flu'], 4)
```




شكل 2.12: تمثيل الإصدار الثالث

قد تلاحظ أنه على الرغم من أن الأعراض الثلاثة على المريض 1: Headache (الصداع)، وTiredness (الإعياء)، وCough (السعال) تظهر عند الإصابة بكل من Flu (الإنفلونزا)، وCovid19 (كوفيد- 19). والحساسية، إلّا أنّ الظّاهر في نتائج التّشخيص هي الإنفلونزا فقط. هذا لأن جميع الأعراض الثلاثة شائعة جدًا في قاعدة المعرِفة، مما يؤدي إلى درجة قصوى قدرها 3. وبالمثل، في ظل معاناة المريض الثاني والثالث من الأعراض نفسها، تؤدي مُدخَلات الأوزان المختلفة للأعراض الأكثر والأقل شيوعًا إلى تشخيصات مختلفة. وعلى وجه التحديد يَنتج عن استخدام وزن متساوٍ لنوعين من الأعراض إضافة الإنفلونزا إلى التشخيص.

وزارة التعليم
Ministry of Education
2025 - 1447

## الإصدار 4

يمكن تحسين النظام القائم على القواعد بزيادة كفاءة قاعدة المعرفة وتجربة **دوال تسجيل النقاط** (Scoring Functions) المختلفة. وعلى الرغم من أن ذلك سيؤدي إلى تحسين النظام، إلا أنه سيتطلب الكثير من الوقت والجهد اليدوي. ولحسن الحظ، هناك طريقة آلية لبناء نظام مبني على القواعد يكون ذكيًا بما يكفي لتصميم قاعدة معرفة ودالة تسجيل نقاط خاصة به: باستخدام تعلُّم الآلة. يُطبِّق **تعلُّم الآلة القائم على القواعد** (Rule-Based Machine Learning) خوارزمية تعلُّم لتحديد القواعد المُفيدة تلقائيًا، بدلًا من الحاجة إلى الإنسان لتطبيق المعرفة والخبرات السابقة في المجال لبناء القواعد وتنظيمها يدويًا.

فبدلًا من قاعدة المعرفة ودالة تسجيل النقاط المُصمَّمتان يدويًا، تَتوقَّع خوارزمية تعلُّم الآلة مُدخَلًا واحدًا فقط وهو مجموعة البيانات التاريخيّة للحالات المَرَضيَّة. فالتعلُّم من البيانات مباشرةً يحول دون حدوث المشكلات المرتبطة باكتساب المعرفة الأساسية والتحقق منها. تتكون كل حالة من بيانات أعراض المريض والتشخيص الطبي الذي يمكن أن يقدمه أي خبير بشري مثل الطبيب. وباستخدام مجموعة بيانات التدريب، تتعلّم الخوارزمية تلقائيًا كيف تتنبأ بالتشخيص المُحتمَل لحالة مريض جديد.

```
import pandas as pd # import pandas to load and process spreadsheet-type data

medical_dataset=pd.read_csv('medical_data.csv') # load a medical dataset.

medical_dataset
```

| | fever | cough | tiredness | headache | stuffy nose | runny nose | sneezing | sore throat | diagnosis |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | covid19 |
| **1** | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | covid19 |
| **2** | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | covid19 |
| **3** | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | covid19 |
| **4** | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | covid19 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **1995** | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | common cold |
| **1996** | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | common cold |
| **1997** | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | common cold |
| **1998** | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | common cold |
| **1999** | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | common cold |

في المثال أعلاه، تحتوي مجموعة البيانات على 2,000 حالة مرضية، بحيث تتكون كل حالة من 8 أعراض محتملة: Fever (الحُمى)، وCough (السُّعال)، وTiredness (الإعياء)، وHeadache (الصُّداع)، وStuffy nose (انسداد الأنف)، وRunny nose (رشح الأنف)، وSneezing (العُطاس)، وSore throat (التهاب الحلق). تُرمَّز كل واحدة من هذه الأعراض في عمود ثنائي مُنفصل. العدد الثنائي 1 يشير إلى أن المريض يُعاني من الأعراض بينما العدد الثنائي 0 يشير إلى أن المريض لا يُعاني من الأعراض.


وزارة التعليم
Ministry of Education
2025 - 1447

يحتوي العمود الأخير على تشخيص الخبير البشري، وهناك أربعة تشخيصات محتملة:

Covid19 (كوفيد- 19)، وFlu (الإنفلونزا)، وAllergies (الحساسية)، وCommon cold (نزلات البرد).

يمكنك التحقق من ذلك بسهولة باستخدام المقطع البرمجي التالي بلغة البايثون:

```
set(medical_dataset['diagnosis'])
```

على الرغم من أن هناك العشرات من خوارزميات تعلُّم الآلة المحتملة التي يمكن استخدامها مع مجموعة البيانات هذه، إلا أنك ستستخدِم تلك التي تتبع المنهجية المُستنِدة على منطق **شجرة القرار** (Decision Tree)، كما ستَستخدِم DecisionTreeClassifier (مصنّف شجرة القرار) من مكتبة البايثون **سكليرن** (Sklearn) على وجه التحديد.

```
from sklearn.tree import DecisionTreeClassifier

def diagnose_v4(train_dataset:pd.DataFrame):

    # create a DecisionTreeClassifier
    model=DecisionTreeClassifier(random_state=1)

    # drop the diagnosis column to get only the symptoms
    train_patient_symptoms=train_dataset.drop(columns=['diagnosis'])

    # get the diagnosis column, to be used as the classification target
    train_diagnoses=train_dataset['diagnosis']

    # build a decision tree
    model.fit(train_patient_symptoms, train_diagnoses)

    # return the trained model
    return model
```

يُعدُّ تطبيق البايثون في الإصدار الرابع أقصر وأبسط بكثير من التطبيقات السابقة، فهو ببساطة يقرأ الملف التدريبي، ويستخدمه لبناء نموذج شجرة القرار استنادًا إلى العلاقات بين الأعراض والتشخيصات، ومن ثَمَّ ينتج نموذجًا مخصَّصًا. لاختبار هذا الإصدار بشكل صحيح، ابدأ بتقسيم مجموعة البيانات إلى مجموعتين منفصلتين، واحدة للتدريب، وأخرى للاختبار.

```
from sklearn.model_selection import train_test_split

# use the function to split the data, get 30% for testing and 70% for training.
train_data, test_data = train_test_split(medical_dataset, test_size=0.3,
random_state=1)

#print the shapes (rows x columns) of the two datasets
print(train_data.shape)
print(test_data.shape)
```

```
(1400, 9)
(600, 9)
```

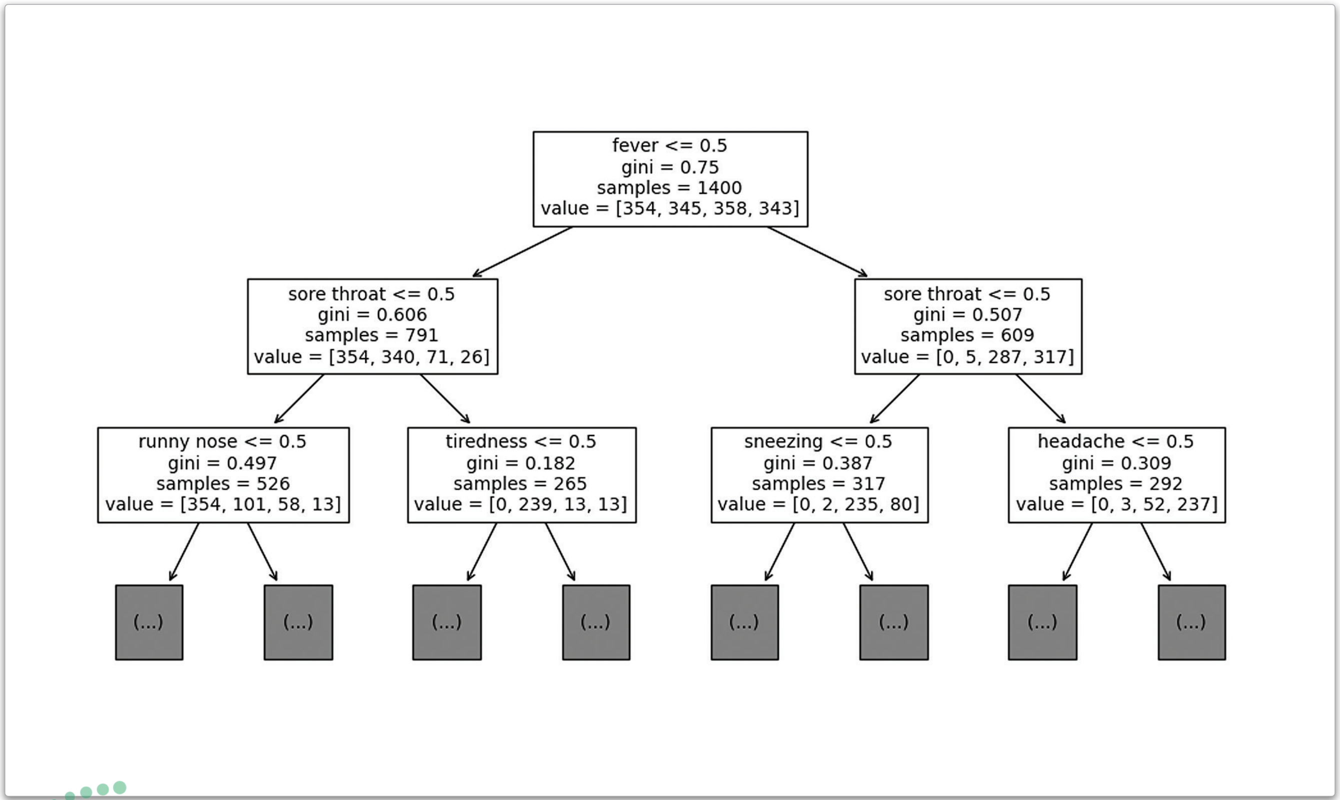
وزارة التعليم
Ministry of Education
2025 - 1447

لديك الآن 1,400 نقطة بيانات ستُستخدَم لتدريب النموذج و600 نقطة ستُستخدَم لاختباره.

ابدأ بتدريب نموذج شجرة القرار وتمثيله:

```
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

my_tree=diagnose_v4(train_data) # train a model

print(my_tree.classes_) # print the possible target labels (diagnoses)

plt.figure(figsize=(12,6)) # size of the visualization, in inches

# plot the tree
plot_tree(my_tree,
          max_depth=2,
          fontsize=10,
          feature_names=medical_dataset.columns[:-1]
         )
```

```
['allergies' 'common cold' 'covid19' 'flu']
```




شكل 2.13: نموذج شجرة القرار لمجموعة بيانات medical_data (البيانات- الطبية) بعُمق مستويين

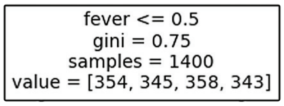

وزارة التعليم
Ministry of Education
2025 - 1447

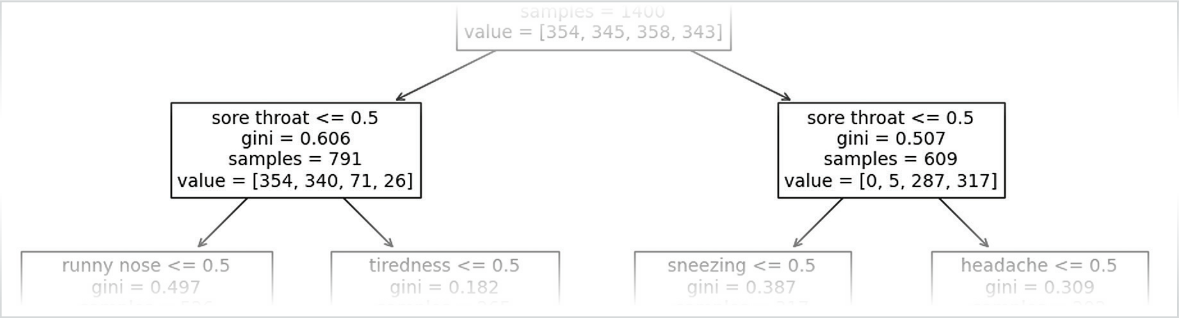
تُستخدَم دالة ( )plot_tree لرسم وعرض شجرة القرار. ولعدم توفر مساحة كافية للعرض سيتم تمثيل المستويين الأوّلين فقط، بالإضافة إلى الجِذر. يمكن ضبط هذا الرقم بسهولة باستخدام المُتغيِّر max_depth.

```
# plot the tree
plot_tree(my_tree,
          max_depth=2,
          fontsize=10
```

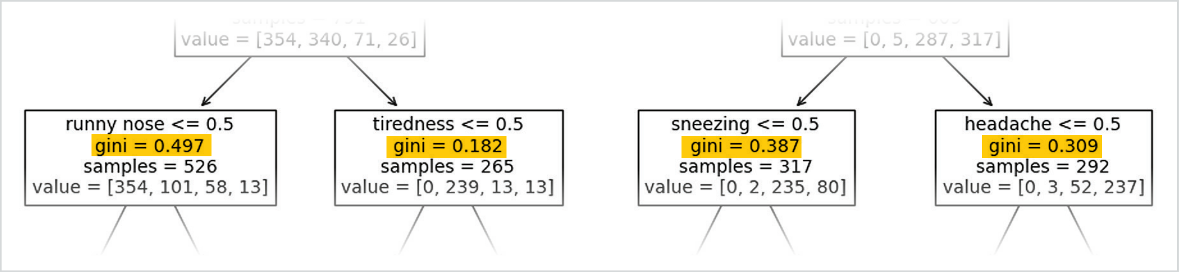
عُمق شجرة القرار.

كل عُقدة في الشجرة تُمثِّل مجموعة فرعية من المرضى، فعلى سبيل المثال، تُمثِّل عُقدة الجِذر إجمالي عدد 1,400 مريض في مجموعة بيانات التدريب. من بينهم، 354، و345، و358، و343 شُخِّصوا بـ Allergies (الحساسية)، وCommon cold (نزلات البرد)، وCovid19 (كوفيد-19)، وFlu (الإنفلونزا)، على التوالي.




بُنِيَّت الشجرة باستخدام نمط من الأعلى إلى الأسفل عبر **التفرُّع الثنائي** (Binary Splits). يَستنِد التفرُّع الأول إلى ما إذا كان المريض يُعاني من الحُمى أم لا. ونظرًا لأن كل خصائص الأعراض ثنائية، يكون التحقق من a <=0.5 صحيحًا إذا لم يكن المريض يعاني من الأعراض. أما المرضى الذين لا يعانون من الحُمى (المسار الأيسر) يتفرَّعون مرة أخرى بناءً على ما إذا كانوا يعانون من التهاب الحلق أم لا. المرضى الذين لا يعانون من التهاب الحلق يتفرَّعون بناءً على ما إذا كانوا يعانون من رشح الأنف أم لا. في هذه المرحلة، تحتوي العُقدة على 526 حالة. تمّ تشخيص 354، و101، و58، و13 من بينهم بالحساسية، ونزلات البرد، وكوفيد-19، والإنفلونزا، على التوالي.




يستمر التفرُّع حتى تُحدِّد الخوارزمية الحالات التي انقسمت بالفعل إلى عُقد نقيَّة تمامًا. العُقدة النقيَّة بالكامل تحتوي على الحالات التي لها التشخيص نفسه. قِيَم مؤشر gini (جيني) المُحدَّدة على كل عُقدة، تُمثِّل مؤشرات على مقياس جيني، وهي صيغة شهيرة تُستخدَم لتقييم درجة نقاء العُقدة.

يقيس مؤشر جيني (Gini Index) الشوائب بالعُقدة، وبالتحديد احتمالية تصنيف محتويات العُقدة بصورة خاطئة. يشير انخفاض مُعامِل جيني إلى ارتفاع درجة تأكُّد الخوارزمية من التصنيف.

وزارة التعليم
Ministry of Education
2025 - 1447

ستَستخدِم الآن شجرة القرار للتنبؤ بالتشخيص الأكثر احتمالًا للمرضى في مجموعة الاختبار.

تُستخدَم مجموعة الاختبار لتقييم أداء النموذج. تَستنِد طريقة التقييم الدقيقة على ما إذا كان المقصود من المهمة **الانحدار** (Regression) أم **التصنيف** (Classification). في مثل مشكلات التصنيف المعروضة هنا، تُستخدَم طرائق التقييم الشهيرة مثل: حساب **دقة النموذج** (Model's Accuracy) و**مصفوفة الدقة** (Confusion Matrix).

- الدقة هي نسبة التنبؤات الصحيحة التي يقوم بها المُصنِّف. تَحقُّق دقة عالية قريبة من 100% يعني أن معظم التنبؤات التي يقوم بها المُصنِّف صحيحة.
- مصفوفة الدقة هي جدول يقارن بين القيم الحقيقية (الفعلية) وبين التنبؤات التي يقوم بها المُصنِّف في مجموعة البيانات. يحتوي الجدول على صف واحد لكل قيمة صحيحة وعمود واحد لكل قيمة مُتوقَّعة. كل مُدخَل في المصفوفة يُمثِّل عدد الحالات التي لها قيم فعلية ومُتوقَّعة.

```
# functions used to evaluate a classifier
from sklearn.metrics import accuracy_score,confusion_matrix

# drop the diagnosis column to get only the symptoms
test_patient_symptoms=test_data.drop(columns=['diagnosis'])

# get the diagnosis column, to be used as the classification target
test_diagnoses=test_data['diagnosis']

# guess the most likely diagnoses
pred=my_tree.predict(test_patient_symptoms)

# print the achieved accuracy score
accuracy_score(test_diagnoses,pred)
```

```
0.8166666666666667
```

ستلاحِظ أن شجرة القرار تُحقِّق دقة تصل إلى 81.6%، وهذا يعني أنه من بين 600 حالة تمّ اختبارها، شَخَّصت الشجرة 490 منها بشكل صحيح. يُمكنك كذلك طباعة مصفوفة الدقة للنموذج لتستعرض بشكل أفضل الأمثلة المُصنَّفة بشكل خاطئ.

```
confusion_matrix(test_diagnoses,pred)
```

```
array([[143,   3,   0,   0],
       [ 48,  98,   5,   4],
       [  2,   1, 127,  12],
       [  1,   3,  31, 122]])
```

| | الحساسية المُتوقَّعة | نزلات البرد المُتوقَّعة | كوفيد19-المُتوقَّع | الإنفلونزا المُتوقَّعة |
|---|---|---|---|---|
| الحساسية الفعلية | **143** | 3 | 0 | 0 |
| نزلات البرد الفعلية | 48 | **98** | 5 | 4 |
| كوفيد-19 الفعلي | 2 | 1 | **127** | 12 |
| الإنفلونزا الفعلية | 1 | 3 | 31 | **122** |

شكل 2.14: مصفوفة الدقة للحالات المُتوقَّعة والحالات الفعلية

الأرقام الواقعة في الخط القُطري (المُظللة باللون الوردي) تُمثِّل الحالات المُتوقَّعة بشكل صحيح، أما الأرقام التي تقع خارج الخط القُطري فتُمثِّل أخطاء النموذج.

على سبيل المثال، بالنظر إلى ترتيب التشخيصات الأربعة المُحتملة [Allergies (الحساسية)، Common cold (نزلات البرد)، Covid19 (كوفيد-19)، Flu (الإنفلونزا)]، توضِّح المصفوفة أن النموذج أخطأ في تصنيف 48 حالة من المُصابين بنزلات البرد بأنهم مصابون بالحساسية، كما أخطأ في تصنيف 31 حالة من المُصابين بالإنفلونزا بأنهم مصابون بكوفيد-19.

وعلى الرغم من أنّ هذا النموذج ليس مثاليًا، فمن المُثير للدهشة أنّه قادر على تحقيق مثل هذه الدرجة العالية من الدقة بتعلُّم مجموعة القواعد الخاصة به، دون الحاجة إلى قاعدة معرِفة أنشئت يدويًا. بالإضافة إلى تحقيق مثل هذه الدقة دون محاولة ضبط مُتغيرات الأداء المتنوعة لـ DecisionTreeClassifier (مُصنِّف شجرة القرار). وبالتالي، يُمكن تحسين دقة النموذج لأفضل من ذلك. كما يُمكن تحسين النموذج بتجاوز قيود النموذج القائم على القواعد وتجربة أنواع مختلفة من خوارزميات تعلُّم الآلة. وستتعلّم بعض هذه الطرائق في الوحدة التالية.

# تمرينات

**1** اذكر بعض مزايا وعيوب الأنظمة القائمة على القواعد.

**2** ما مزايا وعيوب الإصدار الأول؟

**3** أضف إلى المقطع البرمجي الخاص بالإصدار الأول لنظام قائم على القواعد مريضًا يُعاني من الأعراض التالية [Vomiting (القيء)، وAbdominal pain (آلام البطن)، وDiarrhea (الإسهال)، وFever (الحُمى)، وLower back pain (ألم بأسفل الظهر)]. ما التشخيص الطبي لحالة المريض؟ دَوِّن ملاحظاتك بالأسفل.



وزارة التعليم
Ministry of Education
2025 - 1447

4 في الإصدار الثاني، كم عدد الأمراض المُوضحَّة في تشخيص كل مريض إذا غيَّرت قيمة المُتغيِّر matching_symptoms_lower_bound إلى 2 و3 و4؟ عَدِّل المقطع البرمجي ثم دَوِّن ملاحظاتك.

5 في الإصدار الثالث، غَيِّر كلا الوزنين إلى 1 للمريضين الأول والثاني، تمامًا مثل المريض الثالث.

عَدِّل المقطع البرمجي ثم دَوِّن ملاحظاتك.

6 صفْ بإيجاز كيف يُمكن تحسين كل إصدار بالنسبة للإصدار السابق له (الأول إلى الثاني، والثاني إلى الثالث، والثالث إلى الرابع).

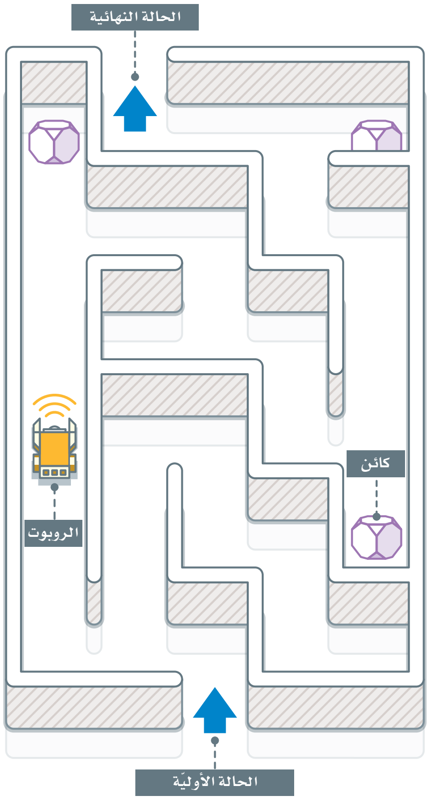
الدرس الرابع
# خوارزميات البحث المستنيرة

رابط الدرس الرقمي
www.ien.edu.sa

## تطبيقات خوارزميات البحث
## Applications of Search Algorithms

خوارزميات البحث هي أحد المكونات الرئيسة لأنظمة الذكاء الاصطناعي، فباستخدامها يُمكن اكتشاف الاحتمالات المختلفة لإيجاد الحلول المناسبة للمشكلات المُعقدة في العديد من التطبيقات السائدة. وفيما يلي أمثلة على بعض تطبيقات خوارزميات البحث:

- **الروبوتية** (Robotics): قد يَستخدِم الروبوت خوارزمية البحث لتحديد طريقه عبر المتاهة أو لتحديد موقع أحد الكائنات في نطاق بيئته.
- **مواقع التجارة الإلكترونية** (E-Commerce Websites): تَستخدِم مواقع التسوق عبر الإنترنت خوارزميات البحث لتُطابق بين استفسارات العملاء وبين المُنتجات المتوفرة، ولتصفية نتائج البحث وفق بعض المعايير مثل: السعر، والعلامة التجارية، والتقييمات، واقتراح المُنتجات ذات الصلة.
- **منصّات مواقع التواصل الاجتماعي** (Social Media Platforms): تَستخدِم مواقع التواصل الاجتماعي خوارزميات البحث لعرض التدوينات، والأشخاص، والمجموعات للمُستخدِمين وفقًا للكلمات المفتاحية واهتمامات المُستخدِم.
- **تمكين الآلة من ممارسة الألعاب بمستوى عالٍ من المهارة** (Enabling a Machine to Play Games at a High Skill Level): يَستخدِم الذكاء الاصطناعي خوارزمية البحث أثناء لعب الشطرنج أو **قو** (Go) لتقييم الحركات المختلفة واختيار الخطوات التي من المرجح أن تؤدي إلى الفوز.
- **نُظم الملاحة باستخدام مُحدِّد المواقع العالمي** (GPS Navigation Systems): تَستخدِم نُظم الملاحة القائمة على مُحدِّد المواقع العالمي خوارزميات البحث لتحديد أقصر وأسرع طريق بين موقعين، مع مراعاة بيانات حركة المرور في الوقت الحالي.
- **نُظم إدارة الملفات** (File Management Systems): تُستخدم خوارزميات البحث في نُظم إدارة الملفات لتحديد موقع الملفات باستخدام اسم، ومحتوى الملف، وبعض السمات الأخرى.



شكل 2.15: استخدام الروبوت خوارزمية البحث لتحديد طريقه

### أنواع خوارزميات البحث وأمثلتها Types and Examples of Search Algorithms

هناك نوعان رئيسان من خوارزميات البحث وهما: **غير المُستنيرة** (Uninformed) **والمُستنيرة** (Informed).

#### خوارزميات البحث غير المُستنيرة Uninformed Search Algorithms

خوارزميات البحث غير المُستنيرة، وتسمّى أيضًا: خوارزميات البحث العمياء، وهي تلك التي لا تحتوي على معلومات إضافية حول حالات المشكلة بإستثناء المعلومات المستفادة من تعريف المشكلة. وتقوم هذه الخوارزميات بإجراء فحص شامل لمساحة البحث استنادًا إلى مجموعة من القواعد المُحدَّدة مُسبقًا. وتُعدُّ تقنيات **البحث بأولوية الاتساع** (BFS) و**البحث بأولوية العمق** (DFS) المُشار إليها في الدرس الثاني أمثلة على خوارزميات البحث غير المُستنيرة.


وزارة التعليم
Ministry of Education
2025 - 1447

على سبيل المثال، تبدأ خوارزمية **البحث بأولوية العمق** (DFS) عند عُقدة الجذر بالشجرة أو المُخطَّط وتتوسَّع حتى تصل للعُقدة الأعمق التي لم تُفحَص. ويستمر الأمر بهذه الطريقة حتى تستنفد الخوارزمية مساحة البحث بأكملها بعد فحص كل العُقد المتاحة. ثم تُخرِج الحل الأمثل الذي وجدته أثناء البحث. فالحقيقة أن خوارزمية **البحث بأولوية العمق** (DFS) تَتبع دومًا هذه القواعد ولا يمكن ضبط استراتيجيتها بصرف النظر عن نتائج البحث، وهذا ما يجعلها خوارزمية غير مُستنيرة.

ومثال آخر ملحوظ على هذا النوع من الخوارزميات هو خوارزمية **البحث بأولوية العمق التكراري المُتعمِّق** (Iterative Deepening Depth-First Search – IDDFS) التي يمكن اعتبارها مزيجًا بين خوارزميتي **البحث بأولوية العمق** (DFS) و**البحث بأولوية الاتساع** (BFS)، فهي تَستخدِم استراتيجية العُمق أولًا للبحث في جميع الخيارات الموجودة في النطاق الكامل بصورة متكررة حتى تصل إلى عُقدة مُحدَّدة.

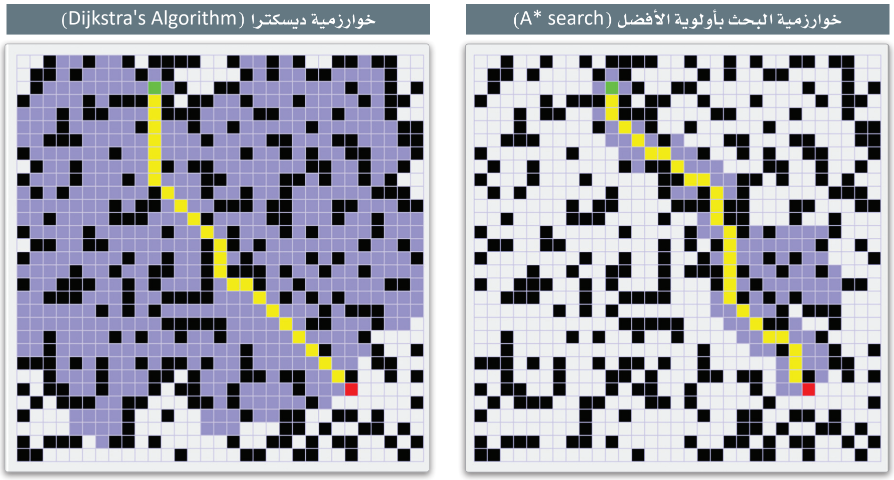
## خوارزميات البحث المُستنيرة Informed Search Algorithms

على النقيض من خوارزميات البحث غير المُستنيرة، تَستخدِم خوارزميات البحث المُستنيرة المعلومات حول المشكلة ومساحة البحث لتوجيه عملية البحث. والأمثلة على هذه الخوارزميات تشمل:

- **خوارزمية البحث بأولوية الأفضل** (A* search) تَستخدِم دالة استدلالية لتقدير المسافة بين كل عُقدة من العُقد المُرشَّحة والعُقدة المُستهدَفة. ثم تُوسِّع العُقدة المُرشَّحة بالتقدير الأقل. إن فعّالية خوارزمية **البحث بأولوية الأفضل** (A* search) مرتبطة بجودة دالتها الاستدلالية. على سبيل المثال، إذا كنت تضمن أن الاستدلال لن يتجاوز المسافة الفعلية إلى الهدف، فبالتالي ستعثر الخوارزمية على الحل الأمثل. بخلاف ذلك، قد لا يكون الحل الناتج من الخوارزمية هو الأفضل.
- **خوارزمية ديكسترا** (Dijkstra's Algorithm) تُوسِّع العُقدة بناء على أقصر مسافة فعلية إلى الهدف في كل خطوة. ولذلك، على النقيض من خوارزمية البحث بأولوية الأفضل، تَحسب خوارزمية **ديكسترا** (Dijkstra) المسافة الفعلية ولا تَستخدِم التقديرات الاستدلالية. وبينما يجعل هذا خوارزمية ديكسترا أبطأ من خوارزمية البحث بأولوية الأفضل، إلا أن ذلك يعني ضمان العثور على الحل الأمثل دومًا (ممثلًا بالمسار الأقصر من البداية حتى الهدف).
- **خوارزمية تسلُّق التلال** (Hill Climbing) تبدأ بتوليد حل عشوائي، ثم تحاول تحسين هذا الحل بصورة متكررة بإجراء تغييرات بسيطة تُحسِّن من دالة استدلالية مُحدَّدة. وبالرغم من أن هذه المنهجية لا تضمن إيجاد الحل الأمثل، إلا أنها سهلة التنفيذ وتتميز بفعالية كبيرة عند تطبيقها على أنواع مُعينة من المشكلات.

**الدالة الاستدلالية (Heuristic Function):**

هي الدالة التي تُصنِّف البدائل في خوارزميات البحث عند كل مرحلة فرعية استنادًا إلى تقديرات استدلالية مبنية على البيانات المتوفرة لتحديد الفرع الذي ستسلكه.




شكل 2.16: حل المتاهة نفسها باستخدام خوارزمية البحث بأولوية الأفضل وخوارزمية ديكسترا

الخلايا ذات اللون البنفسجي هي الخلايا التي تمَّ فحصها، والخلية ذات اللون الأخضر هي موضع البدء، والخلية ذات اللون الأحمر هي موقع الهدف، بينما الخلايا ذات اللون الأصفر تمثل المسار الذي تم العثور عليه.

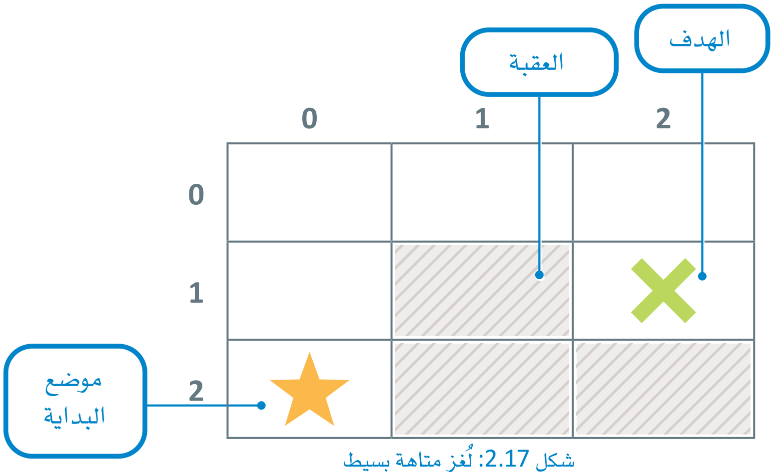
في هذه الوحدة، ستشاهد بعض الأمثلة المرئية وتطبيقات البايثون على خوارزمية **البحث بأولوية الاتساع** (BFS) وخوارزمية **البحث بأولوية الأفضل** (A* search) لمعرفة الاختلافات بين خوارزميتي البحث المُستنيرة وغير المُستنيرة.

## إنشاء ألغاز المتاهة بواسطة البايثون

## Creating Maze Puzzles in Python

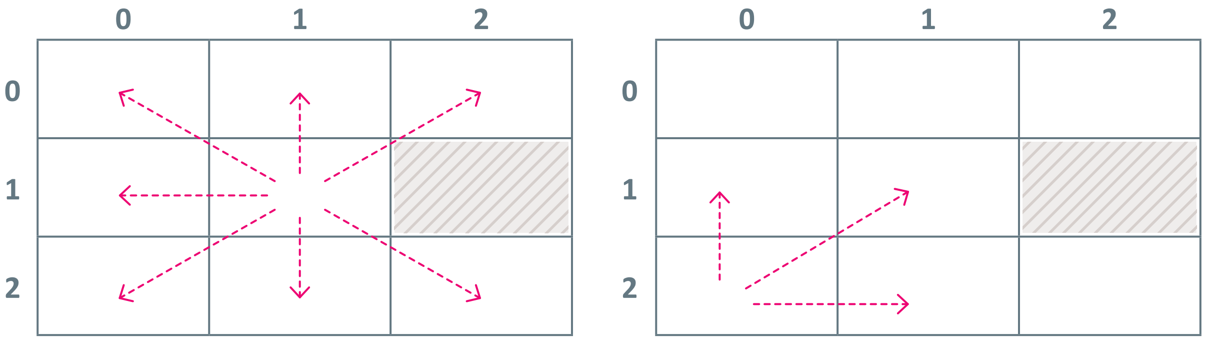
تُعرَّف المتاهة في صورة إطار شبكي 3×3.

يُحدَّد موضع البداية بنجمة في أسفل يسار المتاهة. الهدف هو الوصول إلى الخلية المُستهدَفة المُحدَّدة بالعلامة X، ويمكن للّاعب الانتقال إلى أي خلية فارغة مجاورة لموقعه الحالي.



شكل 2.17: لُغز متاهة بسيط

تكون الخلية فارغة إذا لم تحتوي على عائق. على سبيل المثال، المتاهة الموضَّحة في شكل 2.17 تحتوي على 3 خلايا تشغلُها **الحواجز** (Blocks). هذه الحواجز الملوّنة باللون الرمادي تُشكِّل عائقًا يجب على اللاعب تجاوزه للوصول إلى الهدف X، ويمكن للّاعب الانتقال بشكل أفقي أو رأسي أو قطري إلى أي خلية فارغة مجاورة لموقعه الحالي كما يظهر في الشكل 2.18، على سبيل المثال:


شكل 2.18: يمكن للاعب الانتقال بشكل أفقي أو رأسي أو قطري إلى أي خلية فارغة مجاورة لموقعه الحالي

الهدف هو إيجاد المسار الأقصر والأقلّ عددًا لمرات فحص الخلايا. وبالرغم من أن المتاهة الصغيرة 3×3 قد تبدو بسيطة للّاعب البشري، إلا أنه يتوجب على الخوارزمية الذكية إيجاد حلول للتعامل مع المتاهات الكبيرة والمُعقدة للغاية، مثل: متاهة 10,000×10,000 التي تحتوي على ملايين الحواجز المُوزَّعة في أشكال مُعقَّدة ومتنوعة.

يمكن استخدام المقطع البرمجي التالي بلغة البايثون لإنشاء مجموعة بيانات تُصوّر المثال المُوضَّح في الشكل 2.18.

```
import numpy as np

# create a numeric 3 x 3 matrix full of zeros.
small_maze=np.zeros((3,3))

# coordinates of the cells occupied by blocks
blocks=[(1, 1), (2, 1), (2, 2)]

for block in blocks:
    # set the value of block-occupied cells to be equal to 1
    small_maze[block]=1

small_maze
```

```
array([[0., 0., 0.],
       [0., 1., 0.],
       [0., 1., 1.]])
```

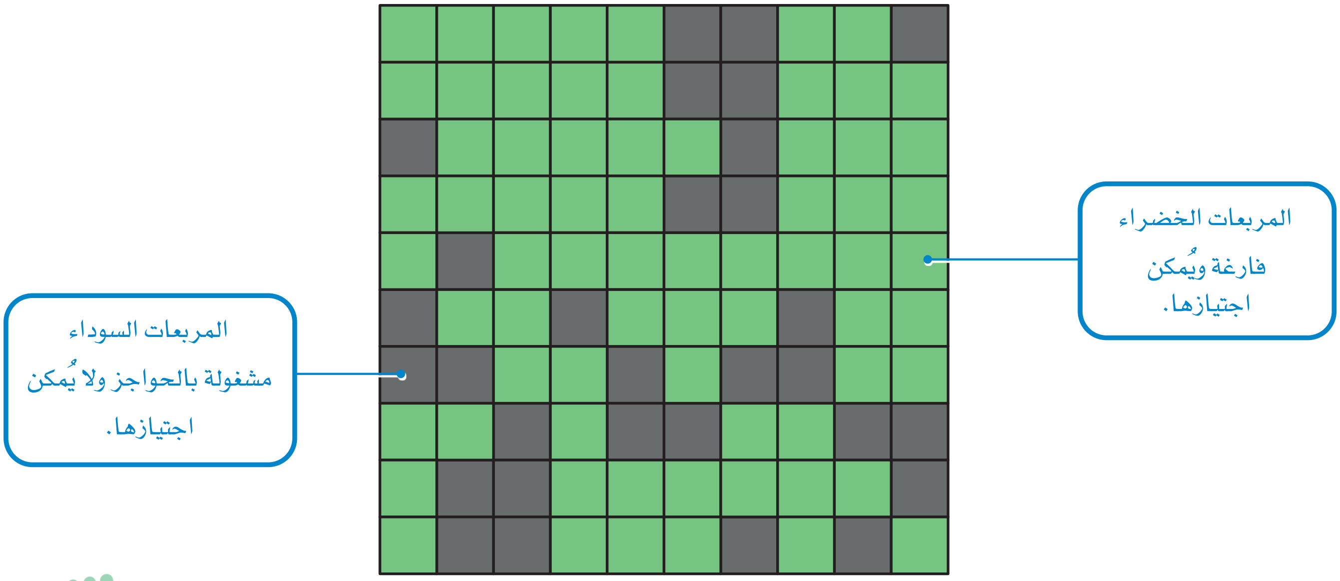

وزارة التعليم
Ministry of Education
2025 - 1447

في هذا التمثيل الرقمي للمتاهة، تُمثَّل الخلايا الفارغة **بالأصفار** (Zeros) والمشغولة **بالآحاد** (Ones). يمكن تحديث المقطع البرمجي نفسه بسهولة لإنشاء متاهات كبيرة ومُعقدة للغاية، مثل:

```
import random

random_maze=np.zeros((10,10))

# coordinates of 30 random cells occupied by blocks
blocks=[(random.randint(0,9),random.randint(0,9)) for i in range(30)]

for block in blocks:
    random_maze[block]=1
```

تُستخدَم الدالة التالية لتمثيل المتاهة:

```
import matplotlib.pyplot as plt # library used for visualization

def plot_maze(maze):
    ax = plt.gca()            # create a new figure
    ax.invert_yaxis()         # invert the y-axis to match the matrix
    ax.axis('off')            # hide the axis labels
    ax.set_aspect('equal') # make sure the cells are rectangular

    plt.pcolormesh(maze, edgecolors='black', linewidth=2,cmap='Accent')
    plt.show()

plot_maze(random_maze)
```




شكل 2.19: تمثيل متاهة 10×10 باستخدام حواجز عشوائية

وزارة التعليم
Ministry of Education
2025 - 1447

يُمكن استخدام الدالة التالية لاستدعاء قائمة تحتوي على كل الخلايا الفارغةِ والمُجاورة لخلية مُحدَّدة في أي متاهة:

```
def get_accessible_neighbors(maze:np.ndarray, cell:tuple):

    # list of accessible neighbors, initialized to empty
    neighbors=[]

    x,y=cell

    # for each adjacent cell position
    for i,j in [(x-1,y-1),(x-1,y),(x-1,y+1),(x,y-1),(x,y+1),(x+1,y-1),(x+1,y),
(x+1,y+1)]:

        # if the adjacent cell is within the bounds of the grid and is not occupied by a block
        if i>=0 and j>=0 and i<len(maze) and j<len(maze[0]) and
maze[(i,j)]==0:

            neighbors.append(((i,j),1))

    return neighbors
```

| | | |
|---|---|---|
| x-1, y-1 | x-1, y | x-1, y+1 |
| x, y-1 | **x, y** | x, y+1 |
| x+1, y-1 | x+1, y | x+1, y+1 |

يَفترض هذا التطبيق أن كل عملية انتقال من خلية إلى أخرى مجاورة سواءً أفقيًا أو رأسيًا أو قطريًا يتم بتكلفة مقدارها وِحدَةٌ وَاحِدَة فقط. سيتم إعادة النظر في هذه الفرضية في وقت لاحق من هذا الدرس بعرض حالات أكثر تعقيدًا مع شروط انتقال مُتغيرة.

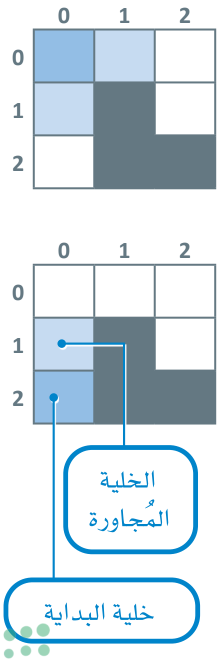
تَستخدِم كل خوارزميات البحث دالة get_accessible_neighbors( ) في محاولة حل المتاهة. في الأمثلة التالية تُستخدم المتاهة 3×3 المُصمَّمة بالأعلى للتحقق من أن الدالة تستدعي الخلية الصحيحة الفارغة والمجاورة للخلية المُحدَّدة.

```
# this cell is the northwest corner of the grid and has only 2 accessible neighbors
get_accessible_neighbors(small_maze, (0,0))
```

```
[((0, 1), 1), ((1, 0), 1)]
```

```
# the starting cell (in the southwest corner) has only 1 accessible neighbor
get_accessible_neighbors(small_maze, (2,0))
```

```
[((1, 0), 1)]
```



شكل 2.20: الخلايا المجاورة

بعد أن تعلّمت كيفية إنشاء المتاهات، وكذلك استدعاء الخلايا المجاورة لأي خلية في المتاهة، فإنّ الخطوة التالية هي تطبيق خوارزميات البحث التي يمكنها حل المتاهة من خلال إيجاد المسار الأقصر من خلية البداية إلى خلية الهدف المُحدَّدة.


وزارة التعليم
Ministry of Education
2025 - 1447

# استخدام خوارزمية البحث بأولوية الاتساع في حل ألغاز المتاهة

## Using BFS to Solve Maze Puzzles

تَستخدِم دالة ( )bfs_maze_solver المُشار إليها في هذا الجزء خوارزمية **البحث بأولوية الاتساع** (BFS) لحل ألغاز المتاهة باستخدام خلية البداية وخلية الهدف. يَستخدِم هذا النموذج دالة ( )get_accessible_neighbors المُحدَّدة بالأعلى لاستدعاء الخلايا المُجاورة التي يمكن فحصها عند أي نقطة أثناء البحث، وبمجرد عثور خوارزمية **البحث بأولوية الاتساع** (BFS) على الخليّة الهدف، ستُستخدم دالة ( )reconstruct_shortest_path المُوضحة بالأسفل لإعادة بِناء المسار الأقصر واستدعائه، وذلك بتتبع المسار بصورة عكسية من خلية الهدف إلى خلية البداية:

```
def reconstruct_shortest_path(parent:dict, start_cell:tuple, target_cell:tuple):

    shortest_path = []

    my_parent=target_cell  # start with the target_cell

    # keep going from parent to parent until the search cell has been reached
    while my_parent!=start_cell:

        shortest_path.append(my_parent) # append the parent

        my_parent=parent[my_parent] # get the parent of the current parent

    shortest_path.append(start_cell) # append the start cell to complete the path

    shortest_path.reverse() # reverse the shortest path

    return shortest_path
```

ستُستخدَم دالة ( )reconstruct_shortest_path أيضًا لإعادة بِناء الحل لخوارزمية **البحث بأولوية الأفضل** (A* search) المُشار إليها سلفًا في هذا الدرس. وبالنظر إلى تعريف الدالتين ( )get_accessible_ neighbors و( )reconstruct_shortest_path المُساعدتين، يُمكن تنفيذ دالة ( )bfs_maze_solver على النحو التالي:

```
from typing import Callable # used to call a function as an argument of another function

def bfs_maze_solver(start_cell:tuple,
                    target_cell:tuple,
                    maze:np.ndarray,
                    get_neighbors: Callable,
                    verbose:bool=False): # by default, suppresses descriptive output text

    cell_visits=0 # keeps track of the number of cells that were visited during the search
    visited = set() # keeps track of the cells that have already been visited
    to_expand = [] # keeps track of the cells that have to be expanded

    # add the start cell to the two lists
    visited.add(start_cell)
    to_expand.append(start_cell)
    # remembers the shortest distance from the start cell to each other cell
    shortest_distance = {}
    # the shortest distance from the start cell to itself, zero
```

```python
    shortest_distance[start_cell] = 0

    # remembers the direct parent of each cell on the shortest path from the start_cell to the cell
    parent = {}
    #the parent of the start cell is itself
    parent[start_cell] = start_cell

    while len(to_expand)>0:

        next_cell = to_expand.pop(0) # get the next cell and remove it from the expansion list

        if verbose:
            print('\nExpanding cell', next_cell)

        # for each neighbor of this cell
        for neighbor,cost in get_neighbors(maze, next_cell):

            if verbose:
                print('\tVisiting neighbor cell',neighbor)

            cell_visits+=1

            if neighbor not in visited: # if this is the first time this neighbor is visited

                visited.add(neighbor)
                to_expand.append(neighbor)
                parent[neighbor]= next_cell
                shortest_distance[neighbor]=shortest_distance[next_cell]+cost

                # target reached
                if neighbor==target_cell:

                    # get the shortest path to the target cell, reconstructed in reverse.
                    shortest_path = reconstruct_shortest_path(parent,
                                                              start_cell, target_cell)

                   return shortest_path, shortest_distance[target_cell],cell_visits

            else: # this neighbor has been visited before

                # if the current shortest distance to the neighbor is longer than the shortest
                # distance to next_cell plus the cost of transitioning from next_cell to this neighbor
                if shortest_distance[neighbor]>shortest_distance[next_cell]
                                                                    +cost:

                    parent[neighbor]=next_cell
                   shortest_distance[neighbor]=shortest_distance[next_cell]+cost

    # search complete but the target was never reached, no path exists
    return None,None,None
```

وزارة التعليم
Ministry of Education
2025 - 1447

تتّبع الدالة منهجية **البحث بأولوية الاتساع** (BFS) للبحث في كل الخيارات في العُمق الحالي قبل الانتقال إلى مستوى العُمق التالي، وتَستخدم هذه المنهجية مجموعة واحدة تُسمّى visited وقائمة تُسمى to_expand.

تتضمن المجموعة الأولى كل الخلايا التي فُحِصَت مرة واحدة على الأقل من قبل الخوارزمية، بينما تتضمن القائمة الثانية كل الخلايا التي لم تُوَسَّع بعد، مما يعني أن الخلايا المُجاورة لم تُفحص بعد. تَستخدم الخوارزمية كذلك قاموسين shortest_distance وparent، يحفظ الأوّل منهما طول المسار الأقصر من خلية البداية إلى كل خلية أخرى، بينما يحفظ الثاني عُقدة الخلية الأصل في المسار الأقصر.

بمجرد الوصول إلى الخلية الهدف وانتهاء البحث، سيُخزِّن المتغيِّر shortest_distance[target_cell] طول الحل والذي يمثل طول المسار الأقصر من البداية إلى الهدف.

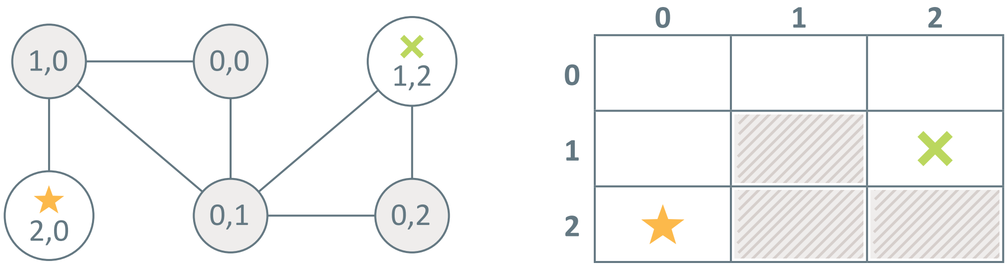
يَستخدِم المقطع البرمجي التالي دالة bfs_maze_solver() لحل المتاهة الصغيرة 3×3 المُوضَّحة بالأعلى:

```
start_cell=(2,0) # start cell, marked by a star in the 3x3 maze
target_cell=(1,2) # target cell, marked by an "X" in the 3x3 maze

solution, distance, cell_visits=bfs_maze_solver(start_cell,
                                                target_cell,
                                                small_maze,
                                                get_accessible_neighbors,
                                                verbose=True)

print('\nShortest Path:', solution)
print('Cells on the Shortest Path:', len(solution))
print('Shortest Path Distance:', distance)
print('Number of cell visits:', cell_visits)
```

```
Expanding cell (2, 0)
        Visiting neighbor cell (1, 0)

Expanding cell (1, 0)
        Visiting neighbor cell (0, 0)
        Visiting neighbor cell (0, 1)
        Visiting neighbor cell (2, 0)

Expanding cell (0, 0)
        Visiting neighbor cell (0, 1)
        Visiting neighbor cell (1, 0)

Expanding cell (0, 1)
        Visiting neighbor cell (0, 0)
        Visiting neighbor cell (0, 2)
        Visiting neighbor cell (1, 0)
        Visiting neighbor cell (1, 2)

Shortest Path: [(2, 0), (1, 0), (0, 1), (1, 2)]
Cells on the Shortest Path: 4
Shortest Path Distance: 3
Number of cell visits: 10
```

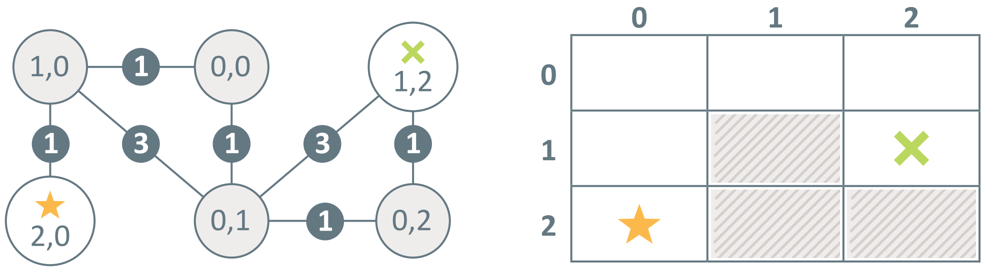
وزارة التعليم
Ministry of Education
2025 - 1447

تنجح خوارزمية **البحث بأولوية الاتساع** (BFS) في إيجاد المسار الأقصر بعد فحص 10 خلايا. يُمكن تصوير عملية البحث المطبّقة بخوارزمية **البحث بأولوية الاتساع** (BFS) بسهولة عند تصوير المتاهة بالتمثيل المُستنِد إلى مُخطَّط. المثال التالي يعرض متاهة 3×3 وتمثيلها بالمُخطَّط:




يتضمن تمثيل المُخطَّط عُقدة واحدة لكل خلية غير مشغولة. تُوضِّح القيمة على العُقد إحداثيات خلية المصفوفة المُقابلة. ستظهر حافة غير مُوجَّهة من عُقدة إلى أخرى في حال كانت الخلايا المُقابِلة يُمكن الوصول إليها من خلال الانتقال من واحدة إلى الأخرى. إحدى الملاحظات المُهمّة حول خوارزمية **البحث بأولوية الاتساع** (BFS) هي أنه في حالة **المُخطَّطات غير الموزونة** (Unweighted Graphs) يكون المسار الأول الذي تُحدِّده الخوارزمية بين خلية البداية وأي خلية أخرى هو المسار الذي يتضمن أقل عدد من الخلايا التي تمّ فحصها. وهذا يعني أنه إذا كانت كلّ الحواف في المُخطَّط لها الوزن نفسه، أي كان لكلّ الانتقالات من خلية إلى أخرى التكلفة نفسها، فإنّ المسار الأول الذي تُحدِّده الخوارزمية إلى عُقدة مُحدَّدة يكون هو المسار الأقصر إلى تلك العُقدة. ولهذا السبب، تتوقف دالة bfs_maze_solver( ) عن البحث، وتَعرض نتيجة المرة الأولى التي فَحصَت فيها العُقدة المُستهدَفة.

ومع ذلك، لا يمكن تطبيق هذه المنهجية على **المُخطَّطات الموزونة** (Weighted Graphs). المثال التالي يوضِّح **إصدارًا موزونًا** (Weighted Version) لتمثيل مُخطَّط متاهة 3×3:




شكل 2.21: المتاهة ومُخطَّطها الموزون

في هذا المثال، يكون وزن كل الحواف المُقابلة للحركات الرأسية أو الأفقية (جنوبًا، شمالًا، غربًا، شرقًا) يساوي 1.

ومع ذلك، يكون وزن كل الحواف المُقابلة للحركات القُطرية (جنوبية غربية، جنوبية شرقية، شمالية غربية، شمالية شرقية) يساوي 3. في هذه الحالة الموزونة، سيكون المسار الأقصر هو [(2,0)، (1,0)، (0,0)، (0,1)، (0,2)، (1,2)]، بمسافة إجمالية: 1+1+1+1+1=5.

يمكن ترميز هذه الحالة الأكثر تعقيدًا باستخدام الإصدار الموزون من الدالة get_accessible_neighbors( ) المُوضَّحة بالأسفل.

```
def get_accessible_neighbors_weighted(maze:np.ndarray,
                                      cell:tuple,
                                      horizontal_vertical_weight:float,
                                      diagonal_weight:float):
```

```python
    neighbors=[]
    x,y=cell

    for i,j in [(x-1,y-1), (x-1,y+1), (x+1,y-1), (x+1,y+1)]: # for diagonal neighbors

        # if the cell is within the bounds of the grid and it is not occupied by a block
        if i>=0 and j>=0 and i<len(maze) and j<len(maze[0]) and maze[(i,j)]==0:

            neighbors.append(((i,j), diagonal_weight))

    for i,j in [(x-1,y), (x,y-1), (x,y+1), (x+1,y)]: # for horizontal and vertical neighbors

        if i>=0 and j>=0 and i<len(maze) and j<len(maze[0]) and maze[(i,j)]==0:

            neighbors.append(((i,j), horizontal_vertical_weight))

    return neighbors
```

تسمح الدالة للمُستخدِم بتعيين وزن مُخصّص للحركات الأفقية والحركات الرأسية، وكذلك وزن مُخصّص مختلف للحركات القُطرية. إذا استُخدِم **الإصدار الموزون** (Weighted Version) المُشار إليه بواسطة **أداة الحل في البحث بأولوية الاتساع** (BFS solver)، فإنّ النتائج ستكون كما يلي:

```python
from functools import partial

start_cell=(2,0)
target_cell=(1,2)
horz_vert_w=1  # weight for horizontal and vertical moves
diag_w=3  # weight for diagonal moves

solution, distance, cell_visits=bfs_maze_solver(start_cell,
                                                target_cell,
                                                small_maze,
                                                partial(get_accessible_neighbors_weighted,
                                                        horizontal_vertical_weight=horz_vert_w,
                                                         diagonal_weight=diag_w),
                                                verbose=False)

print('\nShortest Path:', solution)
print('Cells on the Shortest Path:', len(solution))
print('Shortest Path Distance:', distance)
print('Number of cell visits:', cell_visits)
```

```
Shortest Path: [(2, 0), (1, 0), (0, 1), (1, 2)]
Cells on the Shortest Path: 4
Shortest Path Distance: 7
Number of cell visits: 6
```

وكما هـو مُتوقَّع، أخطـأت **أداة الحل في البحث بأولوية الاتساع** (BFS solver) في عرض المسار السـابق نفسـه بالضبط، على الرغم من أن التكلفة تسـاوي 7، ومن الواضح أنه ليس المسار الأقصر. ويرجع ذلك إلى الطبيعة غير المستنيرة لخوارزمية **البحث بأولوية الاتساع** (BFS)، حيث لا تأخذ الخوارزمية الأوزان بعين الاعتبار عند تحديد الخلية المُقرَّر توسيعها في الخطوة التالية؛ لأنها تُطبِّق ببساطة منهجية البحث بالعرض نفسِها والتي تؤدي إلى المسار نفسه الـذي وجدته الخوارزمية في **الإصدار غير الموزون** (Unweighted Version) مـن المتاهـة. القسـم التـالي يصـف طريقـة معالجـة نقطـة الضعـف هـذه باسـتخدام خوارزميـة **البحث بأولوية الأفضل** (A* search)، وهـي خوارزمية مُستنيرة وأكثر ذكاءً تضبط سلوكها وفقًا لـلأوزان المُحدَّدة، وبالتـالي يُمكنها حل المتاهات باستخدام **الانتقالات الموزونة** (Weighted Transitions) و**الانتقالات غير الموزونة** (Unweighted Transitions).

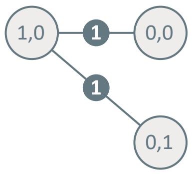
## استخدام خوارزمية البحث بأولوية الأفضل في حل ألغاز المتاهة

## Using A* Search to Solve Maze Puzzles

كما في خوارزمية **البحث بأولوية الاتساع** (BFS)، تَفحَص خوارزمية **البحث بأولوية الأفضل** (A* search) خلية واحدة في كل مرة بفحص كل خلية مجاورة يمكن الوصول إليها. فبينما تستخدِم خوارزمية **البحث بأولوية الاتساع** (BFS) منهجية بحث عمياء بأولوية العرض لتحديد الخلية التالية التي ستَفحَصها، تَفحَص خوارزمية **البحث بأولوية الأفضل** (A* search) الخلية التي يكون بينها وبين الخلية المُستهدَفة أقصر مسافة محسوبة بواسطة **الدالة الاستدلالية** (Heuristic Function). يعتمد التعريف الدقيق للدالة الاستدلالية على التطبيق. في حالة ألغاز المتاهة، توفِّر الدالة الاستدلالية تقديرًا دقيقًا لمدى قُرب الخلية المُرشَّحة إلى الخلية المُستهدَفة. يضمن الاستدلال المُطبَّق عدم **المبالغة في تقدير** (Overestimate) المسافة الفعلية إلى الخلية المُستهدَفة مثل: عرض مسافة تقديرية أكبر من المسافة الحقيقية إلى الهدف، وبالتالي ستُحدِّد الخوارزمية أقصر مسار مُحتمل لكلٍ من المُخطَّطين **الموزون** (Weighted) و**غير الموزون** (Unweighted). إذا كان الاستدلال يُبالِغ في بعض الأحيان في تقدير المسافة، ستُقدِّم خوارزمية **البحث بأولوية الأفضل** (A* search) حلًّا، ولكن قد لا يكون الأفضل. الاستدلال المُحتمل الأبسط الذي لن يؤدي إلى المُبالغة في تقدير المسافة هو دالة بسيطة تعطي دومًا مسافة تقديرية قدرها وحدة واحدة.

```python
def constant_heuristic(candidate_cell:tuple, target_cell:tuple):
    return 1
```

على الرغم من أن هذا الاستدلال شديد التفاؤل، إلا أنه لن يُقدم أبدًا تقديرًا أعلى من المسافة الحقيقية، وبالتالي سيؤدي إلى أفضل حل ممكن. سيتم تقديم استدلال متطور يُمكنه العثور على أفضل حل بشكلٍ سريع في هذا القسم لاحقًا.



شكل 2.22: الاستدلال الثابت

تَستخدِم الدالة التالية دالة استدلالية معطاة للعثور على الخلية التي يجب توسيعها بعد ذلك:

```python
def get_best_candidate(expansion_candidates:set,
                       shortest_distance:dict,
                       heuristic:Callable):

    winner = None
    # best (lowest) distance estimate found so far. Initialized to a very large number
    best_estimate= sys.maxsize

    for candidate in expansion_candidates:

        # distance estimate from start to target, if this candidate is expanded next
```


وزارة التعليم
Ministry of Education
2025 - 1447

```
        candidate_estimate=shortest_distance[candidate]+heuristic(candidate,target_cell)
        if candidate_estimate < best_estimate:

            winner = candidate
            best_estimate=candidate_estimate

    return winner
```

يَستخدِم التطبيق المُشار إليه بالأعلى حلقة التكرار For لفحص الخلايا المُرشَّحة في المجموعة وتحديد الأفضل. ولتطبيق أكثر كفاءة، قد يُستخدم **طابور الأولوية** (Priority Queue) في تحديد المُرشَّح الأفضل دون الحاجة إلى فحص كل المُرشَّحِين بصورة متكررة. تُستخدَم دالة get_best_candidate() كدالةٍ مُساعدة بواسطة دالة astar_maze_solver() المُوضَّحة فيما يلي. وبالإضافة إلى الدالة الاستدلالية، يَستخدِم هذا التطبيق كذلك الدالتين المُساعدتين get_accessible_neighbors_weighted() و reconstruct_shortest_path() المُشار إليهما في القسم السابق.

```
import sys

def astar_maze_solver(start_cell:tuple,
                      target_cell:tuple,
                      maze:np.ndarray,
                      get_neighbors: Callable,
                      heuristic:Callable,
                      verbose:bool=False):

    cell_visits=0

    shortest_distance = {}
    shortest_distance[start_cell] = 0

    parent = {}
    parent[start_cell] = start_cell

    expansion_candidates = set([start_cell])

    fully_expanded = set()

    # while there are still cells to be expanded
    while len(expansion_candidates) > 0:

        best_cell = get_best_candidate(expansion_candidates,shortest_distance,heuristic)

        if best_cell == None:  break

        if verbose: print('\nExpanding cell', best_cell)

        # if the target cell has been reached, reconstruct the shortest path and exit
        if best_cell == target_cell:
```

```
            shortest_path=reconstruct_shortest_path(parent,start_cell,target_cell)

            return shortest_path, shortest_distance[target_cell],cell_visits

        for neighbor,cost in get_neighbors(maze, best_cell):

            if verbose: print('\nVisiting neighbor cell', neighbor)

            cell_visits+=1

            # first time this neighbor is reached
            if neighbor not in expansion_candidates and neighbor not in fully_expanded:

                expansion_candidates.add(neighbor)

                parent[neighbor] = best_cell # mark the best_cell as this neighbor's parent

                # update the shortest distance for this neighbor
                shortest_distance[neighbor] = shortest_distance[best_cell] + cost

            # this neighbor has been visited before, but a better (shorter) path to it has just been found
            elif shortest_distance[neighbor] > shortest_distance[best_cell] + cost:

                shortest_distance[neighbor] = shortest_distance[best_cell] + cost

                parent[neighbor] = best_cell

                if neighbor in fully_expanded:

                    fully_expanded.remove(neighbor)

                    expansion_candidates.add(neighbor)

        # all neighbors of best_cell have been inspected at this point
        expansion_candidates.remove(best_cell)

        fully_expanded.add(best_cell)

    return None, None, None   # no solution was found
```

وكما الحال في الدالة ()bfs_maze_solver، تَستخدِم الدالة المُوضَّحة بالأعلى كذلك القاموسين shortest_distance وparent لحفظ طول المسار الأقصر من خلية البداية إلى أي خلية أخرى، وحفظ عُقدة أصل الخلية في هذا المسار الأقصر.

ورغم ذلك، تَتبع الدالة ()astar_maze_solve منهجية مختلفة لفحص الخلايا وتوسيعها، فهي تَستخدِم expansion_candidates لتتبُّع كل الخلايا التي يمكن توسيعها بعد ذلك. في كل تكرار، تُستخدَم دالة ()get_best_candidate لتحديد أيٍّ من الخلايا المُرشَّحة ستُوسِّعُها بعد ذلك.

بعد اختيار الخليةِ المرشحةِ best_cell، تُستخدَم حلقة التكرار For لفحص كل الخلايا المجاورة لها. إذا كانت الخلية المجاورة تُفحَص للمرة الأولى، فستصبح best_cell عُقدة الأصل للخلية المجاورة في المسار الأقصر.

يحدث الأمر نفسه إذا تم فحص الدالة المجاورة من قبل، ولكن فقط إذا كان المسار إلى هذه الخلية المجاورة من خلال best_cell أقصر من المسار السابق. إذا عثرت الدالة بالفعل على مسار أفضل، فسيتعين على الخلية المجاورة العودة إلى مجموعة expansion_candidates لإعادة تقييم المسار الأقصر إلى الخلايا المجاورة لها. يُستخدم المقطع البرمجي التالي ()astar_maze_solver لحل **الحالة غير الموزونة** (Unweighted Case) للُغز المتاهة 3x3:

```
start_cell=(2,0)
target_cell=(1,2)

solution, distance, cell_visits=astar_maze_solver(start_cell,
                                      target_cell,
                                      small_maze,
                                      get_accessible_neighbors,
                                      constant_heuristic,
                                      verbose=False)

print('\nShortest Path:', solution)
print('Cells on the Shortest Path:', len(solution))
print('Shortest Path Distance:', distance)
print('Number of cell visits:', cell_visits)
```

```
Shortest Path: [(2, 0), (1, 0), (0, 1), (1, 2)]
Cells on the Shortest Path: 4
Shortest Path Distance: 3
Number of cell visits: 12
```

ستبحث **أداة الحل في البحث بأولوية الأفضل** (A* search solver) عن المسار المُحتمل الأقصر والأفضل بعد فحص 12 خلية. وهذا أكثر قليلًا من **أداة الحل في البحث بأولوية الاتساع** (BFS solver) التي وجدت الحل بعد فحص 10 خلايا فقط. هذا يعود إلى بساطة الاستدلال الثابت المُستخدَم لإرشاد ()astar_maze_solver. وكما سيتضح لاحقًا في هذا القسم، يُمكن استخدام دالة استدلال أخرى لتمكين الخوارزمية من إيجاد الحل بشكلٍ أسرع.

الخطوة التالية هي تقييم ما إذا كانت خوارزمية **البحث بأولوية الأفضل** (A* search) قادرة على حل المتاهة الموزونة التي فشلت خوارزمية **البحث بأولوية الاتساع** (BFS) في العثور على أقصر مسار لها أم لا:

```
start_cell=(2,0)
target_cell=(1,2)

horz_vert_w=1  # weight for horizontal and vertical moves
diag_w=3  # weight for diagonal moves

solution, distance, cell_visits=astar_maze_solver(start_cell,
                                    target_cell,
                                    small_maze,
                                    partial(get_accessible_neighbors_weighted,
                                            horizontal_vertical_weight=horz_vert_w,
                                            diagonal_weight=diag_w),
                                    constant_heuristic,
                                    verbose=False)
```

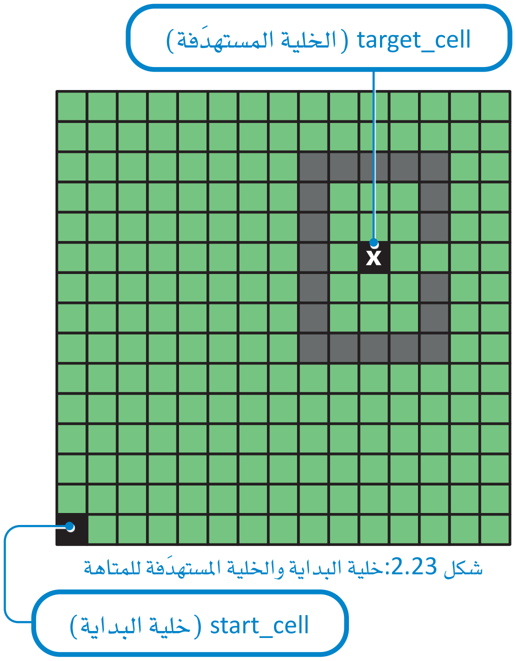
```
print('\nShortest Path:', solution)
print('Cells on the Shortest Path:', len(solution))
print('Shortest Path Distance:', distance)
print('Number of cell visits:', cell_visits)
```

```
Shortest Path: [(2, 0), (1, 0), (0, 0), (0, 1), (0, 2), (1, 2)]
Cells on the Shortest Path: 6
Shortest Path Distance: 5
Number of cell visits: 12
```

تُوضِّح النتائج قدرة ()astar_maze_solver على حل الحالة الموزونة بالعثور على المسار الأقصر المُحتمل [(0،2)، (0،1)، (0،0)، (1،0)، (2،0)، (2،1)] بتكلفة إجمالية قدرها 5. وهذا يوضِّح مزايا استخدام خوارزمية بحث مستنيرة، فهي تُمكِّنك من إيجاد الحل الأمثل باستخدام أبسط طريقة ممكنة.

## المقارنة بين الخوارزميات Algorithm Comparison

الخطوة التالية هي المقارنة بين خوارزمية **البحث بأولوية الاتساع** (BFS) وخوارزمية **البحث بأولوية الأفضل** (A* search) في متاهة أكبر حجمًا وأكثر تعقيدًا. يُستخدَم المقطع البرمجي التالي بلغة البايثون لإنشاء تمثيل رقمي لهذه المتاهة:




شكل 2.23: خلية البداية والخلية المستهدَفة للمتاهة

```
big_maze=np.zeros((15,15))

# coordinates of the cells occupied by blocks
blocks=[(2,8), (2,9), (2,10), (2,11), (2,12),
        (8,8), (8,9), (8,10), (8,11), (8,12),
        (3,8), (4,8), (5,8), (6,8), (7,8),
        (3,12), (4,12), (6,12), (7,12)]

for block in blocks:
    # set the value of block-occupied cells to be equal to 1
    big_maze[block]=1
```

هذه المتاهة 15×15 تحتوي على قسم من الحواجز على شكل حرف C ينبغي على اللّاعب تجاوزها للوصول إلى الهدف المُحدَّد بالعلامة X. ثم تُستخدَم **أداة الحل في البحث بأولوية الاتساع** (BFS solver) و**أداة الحل في البحث بأولوية الأفضل** (A* search solver) لحل الإصدارات الموزونة وغير الموزونة من هذه المتاهة كبيرة الحجم:

**الإصدار غير الموزون**

```
start_cell=(14,0)
target_cell=(5,10)

solution_bfs_unw, distance_bfs_unw, cell_visits_bfs_unw=bfs_maze_solver(start_cell,
                                                          target_cell,
                                                          big_maze,
                                                          get_accessible_neighbors,
```

```
                                        verbose=False)

print('\nBFS unweighted.')
print('\nShortest Path:', solution_bfs_unw)
print('Cells on the Shortest Path:', len(solution_bfs_unw))
print('Shortest Path Distance:', distance_bfs_unw)
print('Number of cell visits:', cell_visits_bfs_unw)

solution_astar_unw, distance_astar_unw, cell_visits_astar_unw=astar_maze_solver(
                                        start_cell,
                                        target_cell,
                                        big_maze,
                                        get_accessible_neighbors,
                                        constant_heuristic,
                                        verbose=False)

print('\nA* Search unweighted with a constant heuristic.')
print('\nShortest Path:', solution_astar_unw)
print('Cells on the Shortest Path:', len(solution_astar_unw))
print('Shortest Path Distance:', distance_astar_unw)
print('Number of cell visits:', cell_visits_astar_unw)
```

```
BFS unweighted.

Shortest Path: [(14, 0), (13, 1), (12, 2), (11, 3), (10, 4), (9, 5), (8,
6), (8, 7), (9, 8), (9, 9), (9, 10), (9, 11), (9, 12), (8, 13), (7, 13),
(6, 13), (5, 12), (4, 11), (5, 10)]
Cells on the Shortest Path: 19
Shortest Path Distance: 18
Number of cell visits: 1237

A* Search unweighted with a constant heuristic.

Shortest Path: [(14, 0), (13, 1), (12, 2), (11, 3), (10, 4), (10, 5), (10,
6), (9, 7), (9, 8), (10, 9), (9, 10), (9, 11), (9, 12), (8, 13), (7, 13),
(6, 13), (5, 12), (6, 11), (5, 10)]
Cells on the Shortest Path: 19
Shortest Path Distance: 18
Number of cell visits: 1272
```

الإصدار الموزون

```
start_cell=(14,0)
target_cell=(5,10)

horz_vert_w=1
diag_w=3

solution_bfs_w, distance_bfs_w, cell_visits_bfs_w=bfs_maze_solver(start_cell,
                                    target_cell,
```

```
                                        big_maze,
                                        partial(get_accessible_neighbors_weighted,
                                                horizontal_vertical_weight=horz_vert_w,
                                                diagonal_weight=diag_w),
                                        verbose=False)

print('\nBFS weighted.')
print('\nShortest Path:', solution_bfs_w)
print('Cells on the Shortest Path:', len(solution_bfs_w))
print('Shortest Path Distance:', distance_bfs_w)
print('Number of cell visits:', cell_visits_bfs_w)

solution_astar_w, distance_astar_w, cell_visits_astar_w=astar_maze_solver(start_cell,
                                        target_cell,
                                        big_maze,
                                        partial(get_accessible_neighbors_weighted,
                                                horizontal_vertical_weight=horz_vert_w,
                                                diagonal_weight=diag_w),
                                        constant_heuristic,
                                        verbose=False)

print('\nA* Search weighted with constant heuristic.')
print('\nShortest Path:', solution_astar_w)
print('Cells on the Shortest Path:', len(solution_astar_w))
print('Shortest Path Distance:', distance_astar_w)
print('Number of cell visits:', cell_visits_astar_w)
```

```
BFS weighted.

Shortest Path: [(14, 0), (14, 1), (14, 2), (13, 2), (13, 3), (12, 3), (12,
4), (11, 4), (11, 5), (10, 5), (10, 6), (9, 6), (9, 7), (9, 8), (9, 9), (9,
10), (9, 11), (9, 12), (9, 13), (8, 13), (7, 13), (6, 13), (5, 13), (5,
12), (4, 11), (5, 10)]
Cells on the Shortest Path: 26
Shortest Path Distance: 30
Number of cell visits: 1235

A* Search weighted with constant heuristic.

Shortest Path: [(14, 0), (13, 0), (12, 0), (11, 0), (10, 0), (9, 0), (9,
1), (9, 2), (9, 3), (9, 4), (9, 5), (9, 6), (9, 7), (9, 8), (9, 9), (9,
10), (9, 11), (9, 12), (9, 13), (8, 13), (7, 13), (6, 13), (5, 13), (5,
12), (5, 11), (5, 10)]
Cells on the Shortest Path: 26
Shortest Path Distance: 25
Number of cell visits: 1245
```


وزارة التعليم
Ministry of Education
2025 - 1447

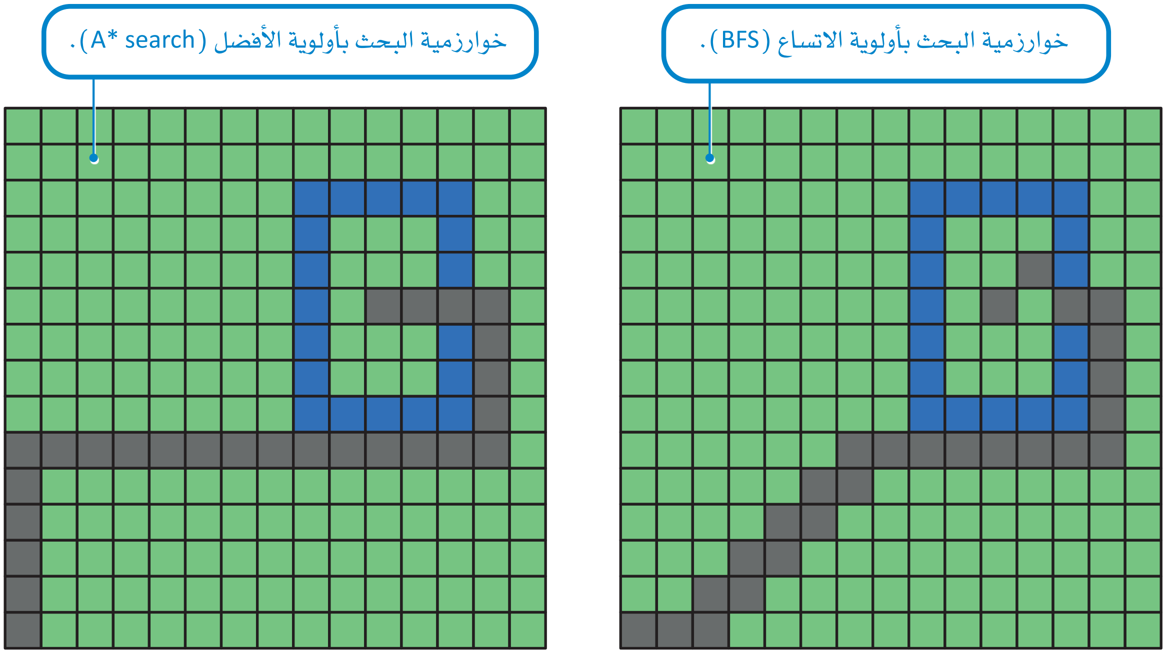
تتوافق النتائج مع تلك التي حصلت عليها في المتاهة الصغيرة وهي كالتالي:

- نجحت خوارزميّتا البحث بأولوية الاتساع (BFS) والبحث بأولوية الأفضل (A* search) في العثور على المسار الأقصر للإصدار غير الموزون.
- وجدت خوارزمية **البحث بأولوية الاتساع** (BFS) الحل بعد فحص عدد أقلّ من الخلايا وهو 1237 مقابل 1272 في خوارزمية **البحث بأولوية الأفضل** (A* search).
- فشلت خوارزمية **البحث بأولوية الاتساع** (BFS) في العثور على المسار الأقصر للإصدار الموزون، حيث عثرت على مسار بطول 30 وحدة.
- عثرت خوارزمية **البحث بأولوية الأفضل** (A* search) على المسار الأقصر للإصدار الموزون، حيث عثرت على مسار بطول 25 وحدة.

يُستخدَم المقطع التالي لتمثيل المسار الأقصر الذي وجدته الخوارزميتان؛ خوارزمية **البحث بأولوية الاتساع** (BFS) وخوارزمية **البحث بأولوية الأفضل** (A* search) للإصدار الموزون كالتالي:

```
maze_astar_w=big_maze.copy()

for cell in solution_astar_w:
    maze_astar_w[cell]=2

plot_maze(maze_astar_w)
```

```
maze_bfs_w=big_maze.copy()

for cell in solution_bfs_w:
    maze_bfs_w[cell]=2

plot_maze(maze_bfs_w)
```




شكل 2.24: مقارنة بين حلَّي خوارزميتي البحث بأولوية الاتساع والبحث بأولوية الأفضل

يؤكد التّمثيلان أن الطبيعة المُستنيرة لخوارزمية **البحث بأولوية الأفضل** (A* search) تسمح لها بتجنب الحركة القُطرية؛ لأن تكلفتها أعلى من الحركتين الأفقية والرأسية. ومن ناحية أخرى، تتجاهل خوارزمية **البحث بأولوية الأفضل** (BFS) غير المُستنيرة تكلفة كل حركة وتُعطي حلًا أعلى تكلفة. وفيما يلي مقارنة عامة بين الخوارزميات المُستنيرة وغير المُستنيرة كما هو موضَّح في الجدول 2.6:

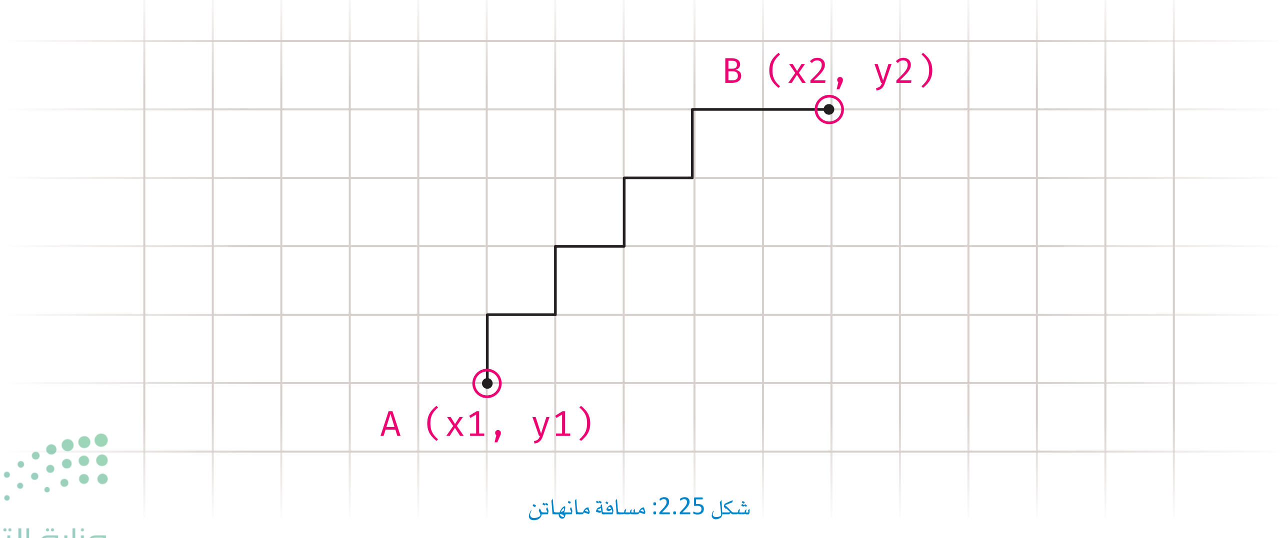
## جدول 2.6: مقارنة بين الخوارزميات المُستنيرة وغير المُستنيرة

| معايير المقارنة | المُستنيرة | غير المُستنيرة |
|---|---|---|
| التعقيد الحسابي (Computational Complexity) | أقل تعقيدًا. | أكثر تعقيدًا حسابيًا. |
| الكفاءة (Efficiency) | أسرع في عمليات البحث. | أبطأ من الخوارزميات المُستنيرة. |
| الأداء (Performance) | أفضل في حل مشكلات البحث واسع النطاق. | غير عملية لحل مشكلات البحث واسع النطاق. |
| الفعالية (Effectiveness) | تحقّقُ حلولا مُناسبةً بشكلٍ عام. | تُحققِ الحل الأمثل. |

ومع ذلك، تُظهِر النتائج أن خوارزمية **البحث بأولوية الاتساع** (BFS) يمكنها العثور على الحل الأمثل بشكلٍ سريع بفحص عدد أقل من الخلايا في الحالة غير الموزونة. يمكن مُعالجة ذلك بتوفير استدلال أكثر ذكاءً لخوارزمية **البحث بأولوية الأفضل** (A* search). والاستدلال الشهير في التطبيقات المُستنِدة إلى المسافة هو **مسافة مانهاتن** (Manhattan Distance)، وهي مجموع الفروقات المُطلقة بين إحداثيَّي نقطتين مُعطاتين. يوضِّح الشكل أدناه مثالًا على كيفية حساب مسافة مانهاتن:

### مسافة مانهاتن Manhattan Distance

```
Manhattan (A, B) = |x1-x2| + |y1-y2|
```




شكل 2.25: مسافة مانهاتن



وزارة التعليم
Ministry of Education
2025 - 1447

يمكن تطبيق هذا بسهولة في صورة دالة البايثون كما يلي:

```
def manhattan_heuristic(candidate_cell:tuple,target_cell:tuple):

    x1,y1=candidate_cell
    x2,y2=target_cell
    return  abs(x1 - x2) + abs(y1 - y2)
```

يُستخدَم المقطع البرمجي التالي لاختبار إمكانية استخدام هذا الاستدلال الذكي لدعم ( )astar_maze_solver في البحث بشكل أسرع في كلٍ من الحالات الموزونة وغير الموزونة:

```
start_cell=(14,0)
target_cell=(5,10)

solution_astar_unw_mn, distance_astar_unw_mn, cell_visits_astar_unw_mn=astar_
maze_solver(start_cell,
            target_cell,
            big_maze,
            get_accessible_neighbors,
            manhattan_heuristic,
            verbose=False)

print('\nA* Search unweighted with the Manhattan heuristic.')
print('\nShortest Path:', solution_astar_unw_mn)
print('Cells on the Shortest Path:', len(solution_astar_unw_mn))
print('Shortest Path Distance:', distance_astar_unw_mn)
print('Number of cell visits:', cell_visits_astar_unw_mn)

horz_vert_w=1  # weight for horizontal and vertical moves
diag_w=3  # weight for diagonal moves

solution_astar_w_mn, distance_astar_w_mn, cell_visits_astar_w_mn=astar_maze_
solver(start_cell,
       target_cell,
       big_maze,
       partial(get_accessible_neighbors_weighted,
               horizontal_vertical_weight=horz_vert_w,
               diagonal_weight=diag_w),
       manhattan_heuristic,
       verbose=False)

print('\nA* Search weighted with the Manhattan heuristic.')
print('\nShortest Path:', solution_astar_w_mn)
print('Cells on the Shortest Path:', len(solution_astar_w_mn))
print('Shortest Path Distance:', distance_astar_w_mn)
print('Number of cell visits:', cell_visits_astar_w_mn)
```

```
A* Search unweighted with the Manhattan heuristic.

Shortest Path: [(14, 0), (13, 1), (12, 2), (11, 3), (10, 4), (9, 5), (8,
6), (8, 7), (9, 8), (9, 9), (9, 10), (9, 11), (9, 12), (8, 13), (7, 13),
(6, 13), (5, 12), (5, 11), (5, 10)]
Cells on the Shortest Path: 19
Shortest Path Distance: 18
Number of cell visits: 865

A* Search weighted with the Manhattan heuristic.

Shortest Path: [(14, 0), (14, 1), (13, 1), (12, 1), (12, 2), (12, 3), (12,
4), (12, 5), (12, 6), (12, 7), (11, 7), (11, 8), (10, 8), (9, 8), (9, 9),
(9, 10), (9, 11), (9, 12), (9, 13), (8, 13), (7, 13), (6, 13), (5, 13), (5,
12), (5, 11), (5, 10)]
Cells on the Shortest Path: 26
Shortest Path Distance: 25
Number of cell visits: 1033
```

تؤكد النتائج أن استدلال **مسافة مانهاتن** (Manhattan Distance) يمكن استخدامه لدعم خوارزمية **البحث بأولوية الأفضل** (A* search) في العثور على المسارات الأقصر المُحتمَلة بفحص أقل عدد من الخلايا في كلٍ من الحالات الموزونة وغير الموزونة. علمًا بأن استخدام هذا الاستدلال الأكثر ذكاءً يفحص عددًا أقل من الخلايا من ذلك المُستخدَم في خوارزمية **البحث بأولوية الاتساع** (BFS).

يُلخِّص الجدول 2.7 النتائج حول مُتغيرات الخوارزميّات المختلفة في المتاهة الكبيرة:

## جدول 2.7: مقارنة بين أداء الخوارزميات

| | خوارزمية البحث بأولوية الاتساع (BFS) | خوارزمية البحث بأولوية الأفضل (A* search) بالاستدلال الثابت | خوارزمية البحث بأولوية الأفضل (A* search) باستدلال مانهاتن |
|---|---|---|---|
| الموزونة | المسافة= 30، وفَحصَت 1235 | المسافة= 25، وفَحصَت 1245 | المسافة= 25، وفَحصَت 1033 |
| غير الموزونة | المسافة= 18، وفَحصَت1237 | المسافة= 18، وفَحصَت 1272 | المسافة= 18، وفَحصَت865 |

يُوضِّح الجدول مزايا استخدام الطَّرائق الأكثر ذكاءً لحل المشكلات المُستنِدة إلى البحث مثل تلك المُوضَّحة بهذا الدرس:

- التَحوُّل من خوارزمية **البحث بأولوية الاتساع** (BFS) غير المُستنيرة إلى خوارزمية **البحث بأولوية الأفضل** (A* search) المُستنيرة حَقَّق نتائجَ أفضل، كما أتاح إمكانية حل المشكلات الأكثر تعقيدًا.
- يُمكن تحسين ذكاء خوارزميات البحث المُستنيرة باستخدام دوال الاستدلال الأفضل التي تسمح لها بالعثور على الحل الأمثل بشكلٍ أسرع.


وزارة التعليم
Ministry of Education
2025 - 1447

# تمرينات

1 اذكر تطبيقين لخوارزميات البحث.

2 حدِّد الاختلافات بين خوارزميات البحث المُستنيرة وغير المُستنيرة، ثم اذكر مثالًا على كل خوارزمية.

وزارة التعليم
Ministry of Education
2025 - 1447

3 اشرح بإيجاز كيف تعمل خوارزمية البحث بأولوية الأفضل (A* search).

4 عَدِّل المقطع البرمجي بتغيير الوزن القُطري (Diagonal Weight) من 3 إلى 1.5. ماذا تُلاحظ؟
هل يتغير المسار الأقصر في حالتي خوارزمية البحث بأولوية الاتساع (BFS) وخوارزمية البحث بأولوية الأفضل (A* search)؟

5 عَدِّل المقطع البرمجي بتبديل إحداثيات خلية البداية مع احداثيات الخلية المُستهدَفة. ماذا تُلاحظ؟
هل المسار هو نفسه كما كان سابقًا للحالات الموزونة من خوارزميتي البحث بأولوية الاتساع (BFS) والبحث بأولوية الأفضل (A* search)؟


وزارة التعليم
Ministry of Education
2025 - 1447

# المشروع

1 عدِّل المقطع البرمجي لخوارزمية **البحث بأولوية الاتساع** (BFS) وخوارزمية **البحث بأولوية الأفضل** (A* search) الموزونتين بتغيير الأوزان الأفقية والرأسية إلى 3 والأوزان القُطرية إلى 5، وكذلك عدِّل نقطة البداية إلى (2، 7).

2 ما المسار الجديد ذو المسافة الأقصر، وما عدد الخلايا التي فُحِصَت في الإصدارات غير الموزونة لخوارزميتي **البحث بأولوية الاتساع** (BFS) و**البحث بأولوية الأفضل** (A* search) باستخدام دالة الاستدلال الثابت؟ حَدِّد هذه القيم ودوِّن ملاحظاتك.

3 اتبع الخطوات نفسها للإصدارات الموزونة من خوارزميتي **البحث بأولوية الاتساع** (BFS) و**البحث بأولوية الأفضل** (A* search) باستخدام دالة الاستدلال الثابت.

4 كرِّر العملية للإصدارات غير الموزونة والموزونة من خوارزميتي **البحث بأولوية الاتساع** (BFS) و**البحث بأولوية الأفضل** (A* search) باستخدام دالة **استدلال مانهاتن** (Manhattan Heuristic).

وزارة التعليم
Ministry of Education
2025 - 1447

# ماذا تعلّمت

- استخدام الاستدعاء الذاتي لحل المشكلات.
- تطبيق خوارزميات اجتياز المُخطَّط المُتقدمة.
- تطبيق الأنظمة القائمة على القواعد البسيطة والمتقدمة.
- تصميم نموذج الذكاء الاصطناعي.
- قياس فعالية نموذج الذكاء الاصطناعي الذي صمَّمتَه.
- استخدام خوارزميات البحث لمحاكاة حلّ مشكلات الحياة الواقعية.

## المصطلحات الرئيسة

| | |
|---|---|
| Model Training | تدريب النموذج |
| Path Finding | إيجاد المسار |
| Recursion | الاستدعاء الذاتي |
| Rule-Based Systems | الأنظمة القائمة على القواعد |
| Scoring Function | دالة تسجيل النقاط |
| Search Algorithms | خوارزميات البحث |
| Uninformed Search | البحث غير المُستنير |
| Unweighted Graph | مُخطَّط غير موزون |
| Weighted Graph | مُخطَّط موزون |

| | |
|---|---|
| A* Search | البحث بأولوية الأفضل |
| Algorithm Performance | أداء الخوارزمية |
| Breadth-First Search (BFS) | البحث بأولوية الاتساع |
| Confusion Matrix | مصفوفة الدقة |
| Depth-First Search (DFS) | البحث بأولوية العمق |
| Heuristic Function | دالة استدلالية |
| Informed Search | البحث المُستنير |
| Knowledge Base | قاعدة المعرِفة |
| Maze Solving | حل المتاهات |

وزارة التعليم
Ministry of Education
2025 - 1447

# 3. معالجة اللغات الطبيعية

سيتعلّم الطالب في هذه الوحدة عملية تدريب شاملة لنموذج التعلُّم الموجَّه والتعلُّم غير الموجَّه لفهم المعنى الكامن في أجزاء النصوص. وكذلك سيتعلّم كيفية استخدام تعلُّم الآلة (Machine Learning - ML) في دعم التطبيقات ذات الصلة بمعالجة اللغات الطبيعية (Natural Language Processing - NLP).

## أهداف التعلُّم

بنهاية هذه الوحدة سيكون الطالب قادرًا على أن:

- يُعرِّف التعلُّم الموجَّه.
- يُدرِّب نموذج التعلُّم الموجَّه على فهم النص.
- يُعرِّف التعلُّم غير الموجَّه.
- يُدرِّب نموذج التعلُّم غير الموجَّه على فهم النص.
- يُنشئ روبوت دردشة بسيط.
- يُنتج النصوص باستخدام تقنيات توليد اللغات الطبيعية (Natural Language Generation - NLG).

## الأدوات

- مفكّرة جوبيتر (Jupyter Notebook)

وزارة التعليم
Ministry of Education
2025 - 1447

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الأول

# التعلُّم الموجَّه

## استخدام التعلُّم الموجَّه لفهم النصوص
## Using Supervised Learning to Understand Text

**معالجة اللغات الطبيعية** (Natural Language Processing - NLP) هي إحدى مجالات **الذكاء الاصطناعي** (Artificial Intelligence - AI) التي تركّز على تمكين أجهزة الحاسب لتصبح قادرة على فهم اللغات البشريّة، وتفسيرها، وإنتاجها. حيث تُعنى معالجة اللغات الطبيعية بعدد من المهام، مثل: تصنيف النصوص، وتحليل المشاعر، والترجمة الآلية، والإجابة على الأسئلة. سيركز هذا الدرس بشكل خاص على كيفية استخدام التعلُّم الموجَّه الذي يُعدُّ أحد الأنواع الرئيسة **لتعلُّم الآلة** (Machine Learning - ML) في تحقيق الفهم والتنبؤ التلقائي لخصائص النصوص.

لقد تعلّمت في الوحدة الأولى أن الذكاء الاصطناعي هو مصطلح يشملُ كلًّا من تعلُّم الآلة والتعلُّم العميق، كما يتضح في الشكل 3.1، فالذكاء الاصطناعي هو ذلك المجال الواسع من علوم الحاسب الذي يُعنى بابتكار آلات ذكية، بينما تعلُّم الآلة هو أحد فروع الذكاء الاصطناعي الذي يركّز على تصميم الخوارزميات وبناء النماذج التي تُمكِّن الآلة من التعلُّم من البيانات دون الحاجة إلى برمجتها بشكل صريح.

شكل 3.1: فروع الذكاء الاصطناعي

**التعلُّم العميق (Deep Learning):**

التعلُّم العميق هو أحد أنواع تعلُّم الآلة الذي يستخدِم الشبكات العصبية العميقة للتعلُّم تلقائيًا من مجموعات كبيرة من البيانات، فهو يسمح لأجهزة الحاسب بالتعرّف على الأنماط واتخاذ القرارات بطريقة تحاكي الإنسان، عبر تصميم نماذج مُعقدة من البيانات.

### تعلُّم الآلة Machine Learning

تعلُّم الآلة هو أحد فروع الذكاء الاصطناعي المعني بتطوير الخوارزميات التي تُمكِّن أجهزة الحاسب من التعلُّم من البيانات المُدخَلة، بدلًا من اتباع التعليمات البرمجية الصريحة، فهو يعمل على تدريب نماذج الحاسب للتعرُّف على الأنماط والقيام بالتنبؤات وفقًا للبيانات المُدخَلة مما يسمح للنموذج بتحسين الدقة مع مرور الوقت، وكذلك يتيح للآلة أداء مهام متعددة، مثل: التصنيف، والانحدار، والتجميع، وتقديم التوصيات دون الحاجة إلى برمجة الآلة بشكل صريح للقيام بكل مُهِمَّة على حدة. يمكن تصنيف تعلُّم الآلة إلى ثلاثة أنواع رئيسة:

**التعلُّم الموجَّه (Supervised Learning)** هو نوع من تعلُّم الآلة تتعلّم فيه الخوارزمية من بيانات تدريب **مُعنوَنة** (Labelled) بهدف القيام بالتنبؤات حول بيانات جديدة غير موجودة في مجموعة التدريب أو الاختبار كما هو موضَّح في الشكل 3.2، ومن الأمثلة عليه:

- **تصنيف الصور** (Image Classification)، مثل: التعرف على الكائنات في الصور.
- **كشف الاحتيال** (Fraud Detection)، مثل: تحديد المُعامَلات المالية المشبوهة.
- **تصفية البريد الإلكتروني العشوائي** (Spam Filtering)، مثل: تحديد رسائل البريد الإلكتروني غير المرغوب فيها.

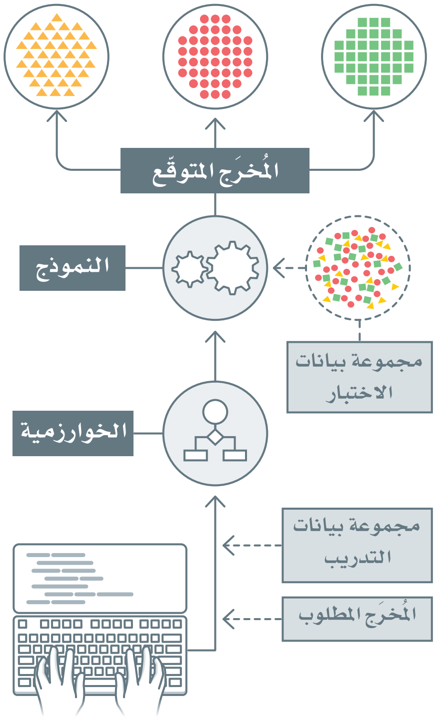
**التعلُّم غير الموجَّه (Unsupervised Learning)** هو نوع من تعلُّم الآلة تعمل فيه الخوارزمية بموجب بيانات **غير مُعنوَنة** (Unlabeled) في محاولة لإيجاد الأنماط والعلاقات بين البيانات، ومن الأمثلة عليه:

- **الكشف عن الاختلاف** (Anomaly Detection)، مثل: تحديد الأنماط غير العادية في البيانات.
- **التجميع** (Clustering)، مثل: تجميع البيانات ذات الخصائص المتشابهة.
- **تقليص الأبعاد** (Dimensionality Reduction)، مثل: اختيار الأبعاد المُستخدَمة للحدِّ من تعقيد البيانات.

**التعلُّم المعزَّز (Reinforcement Learning)** هو نوع من تعلُّم الآلة تتفاعل فيه الآلة مع البيئة المحيطة وتتعلّم عبر المحاولة والخطأ أو تلقّي المكافأة والعقاب، ومن الأمثلة عليه:

- لعب الألعاب، مثل: لعبة الشطرنج أو لعبة **قو** (GO).
- الروبوتية، مثل: تعليم الروبوت كيف يتنقل في البيئة المحيطة به.
- تخصيص الموارد، مثل: تحسين استخدام الموارد في شبكة ما.

جدول 3.1 يلخص مزايا أنواع تعلُّم الآلة وعيوبها.



شكل 3.2: تمثيل التعلُّم الموجَّه

**جدول 3.1: مزايا أنواع تعلُّم الآلة، وعيوبها**

| المزايا | العيوب |
|---|---|
| **التعلُّم الموجَّه** | |
| • أثبت كفاءة وفعالية كبيرة ويُستخدم على نطاق واسع.<br>• سهل الفهم والتطبيق.<br>• يُمكنه التعامل مع البيانات الخطية وغير الخطية على حد سواء. | • يتطلب بيانات مُعنوَنة، والتي قد تكون مرتفعة التكلفة.<br>• يقتصر استخدامه على المُهمَّة التي تم تدريبه عليها، وقد لا يمكنه إعطاء التنبؤ الصحيح للبيانات الجديدة.<br>• يصعب تكيفه مع المشكلات الأخرى في حالات النماذج المُعقدة جدًا. |
| **التعلُّم غير الموجَّه** | |
| • لا يتطلب بيانات مُعنوَنة، مما يجعله أكثر مرونة.<br>• يُمكنه اكتشاف الأنماط الخفية في البيانات.<br>• يُمكنه التعامل مع البيانات الضخمة والمُعقدة. | • أصعب من التعلُّم الموجَّه من حيث الفهم والتفسير.<br>• يقتصر على التحليل الاستكشافي، وقد لا يناسب عمليات صُنع القرار.<br>• يصعب تكيفه مع المشكلات الأخرى في حالات النماذج المُعقدة جدًا. |
| **التعلُّم المعزَّز** | |
| • يتّسم بالمرونة، ويُمكنه التعامل مع البيئات المُعقدة والمتغيرة باستمرار.<br>• يمكنه التعلُّم من التجارب السابقة وتحسين الكفاءة مع مرور الوقت.<br>• يتناسب مع عمليات صُنع القرار مثل لعب الألعاب والروبوتية. | • أكثر تعقيدًا من التعلُّم الموجَّه وغير الموجَّه.<br>• صعوبة تصميم نُظم مكافآت تُحدد السلوك المطلوب بشكل دقيق.<br>• قد يتطلب مجموعات كبيرة من بيانات التدريب والموارد الحسابية. |

وزارة التعليم
Ministry of Education
2025 - 1447

## التعلُّم الموجَّه Supervised Learning

التعلُّم الموجَّه هو أحد أنواع تعلُّم الآلة الذي يعتمد على استخدام البيانات المُعنوَنة لتدريب الخوارزميات للقيام بالتنبؤات. يتم تدريب الخوارزمية على مجموعة من البيانات المُعنوَنة ثم اختبارها على مجموعة بيانات جديدة لم تكن جزءًا من بيانات التدريب. يُستخدَم التعلُّم الموجَّه عادةً في معالجة اللغات الطبيعية للقيام بمهام مثل: تصنيف النصوص، وتحليل المشاعر، **والتعرّف على الكيانات المسماة** (Named Entity Recognition – NER). في هذه المهام يتم تدريب الخوارزمية على مجموعة من البيانات المُعنوَنة، حيث يتم إدراج كل مثال تحت عنوان التصنيف المناسب أو المشاعر المناسبة. يُطلَق على عملية التعلُّم الموجَّه اسم **الانحدار** (Regression) عندما تكون القيم التي تتنبأ بها الآلة رقميّة، بينما يطلق عليها اسم **التصنيف** (Classification) عندما تكون القيم متقطّعة.

**التعلُّم الموجَّه (Supervised Learning):**

ستستخدِم في التعلُّم الموجَّه مجموعات البيانات المُعنوَنة والمُنظمة بشكل يدوي لتدريب خوارزميات الحاسب على التنبؤ بالقيم الجديدة.

### الانحدار

على سبيل المثال، قد يُستخدم الانحدار في التنبؤ بسعر بيع المنزل وفقًا لمساحته، وموقعه، وعدد غرف النوم فيه. كما يمكن استخدامه في التنبؤ بحجم الطلب على أحد المنتجات استنادًا إلى بيانات المبيعات التاريخية وحجم الإنفاق الإعلاني. وفي مجال معالجة اللغات الطبيعية، يَستخدِم الانحدار النصوص المُدخَلة المتوفرة للتنبؤ بتقييم الجمهور للفيلم أو مدى التفاعل مع المنشورات الخاصة به على وسائل التواصل الاجتماعي.

### التصنيف

من ناحية أخرى، يُستخدم التصنيف في التطبيقات مثل: تشخيص الحالات الطبية وفقًا للأعراض ونتائج الفحوصات. وعندما يتعلق الأمر بفهم النصوص، يمكن استخدام التعلُّم الموجَّه في تصنيف النصوص المُدخَلة إلى فئات أو عناوين أو التنبؤ بها بناءً على الكلمات أو العبارات الموجودة في المُستنَد. على سبيل المثال، يمكن تدريب نموذج التعلُّم الموجَّه لتصنيف رسائل البريد الإلكتروني إلى رسائل مزعجة أو غير مزعجة وفقًا للكلمات أو العبارات المُستخدَمة في رسالة البريد الإلكتروني. ويُعدّ تصنيف المشاعر أحد التطبيقات الشهيرة كذلك، حيث يمكن التنبؤ بالانطباع العام حول مستند ما سواء كان سلبيًّا أم إيجابيًّا. وسَيُستخدم هذا التطبيق كمثال عملي في هذه الوحدة، لشرح كل خطوات عملية بِناء واستخدام نموذج التعلُّم الموجَّه بشكل شامل من بداية رحلة التعلُّم حتى نهايتها.

في هذه الوحدة ستَستخدم مجموعة بيانات من مراجعات الأفلام على موقع IMDb.com الشهير. ستجد البيانات مُقسّمه إلى مجموعتين؛ الأولى ستُستخدم لتدريب النموذج، والثانية لاختبار أداء النموذج. في البداية لابد أن تُحَمّل البيانات إلى DataFrame، لِذا عليك استخدام مكتبة **بانداس بايثون** (Pandas Python) والتي استخدمتها سابقًا. مكتبة بانداس هي إحدى الأدوات الشهيرة التي تُستخدم للتعامل مع جداول البيانات. التعليمات البرمجية التالية ستقوم باستيراد المكتبة إلى البرنامج، ثم تحميل مجموعتي البيانات:

```
%%capture  # capture is used to suppress the installation output.

# install the pandas library, if it is missing.
!pip install pandas
import pandas as pd
```

مكتبة بانداس هي مكتبة شهيرة تُستخدم لقراءة ومعالجة البيانات الشبيهة بجداول البيانات.



وزارة التعليم
Ministry of Education
2025 - 1447

```python
# load the train and testing data.
imdb_train_reviews=pd.read_csv('imdb_data/imdb_train.csv')
imdb_test_reviews=pd.read_csv('imdb_data/imdb_test.csv')

imdb_train_reviews
```

| | text | label |
|---|---|---|
| **0** | I grew up (b. 1965) watching and loving the Th... | 0 |
| **1** | When I put this movie in my DVD player, and sa... | 0 |
| **2** | Why do people who do not know what a particula... | 0 |
| **3** | Even though I have great interest in Biblical ... | 0 |
| **4** | Im a die hard Dads Army fan and nothing will e... | 1 |
| **...** | ... | ... |
| **39995** | "Western Union" is something of a forgotten cl... | 1 |
| **39996** | This movie is an incredible piece of work. It ... | 1 |
| **39997** | My wife and I watched this movie because we pl... | 0 |
| **39998** | When I first watched Flatliners, I was amazed.... | 1 |
| **39999** | Why would this film be so good, but only gross... | 1 |

40000 rows × 2 columns

تقييم إيجابي

تقييم سلبي

شكل 3.3: مجموعة بيانات التدريب المُعنوَنة

وكما يتضـح في الشـكل 3.3، فـإن مجموعـة بيانـات DataFrame تحتوي على عمودين:

- نصّ التقييم.
- القيم (الصَّنف).

القيمة 0 تمثل تقييمًا سلبيًّا

بينما القيمة 1 تمثل تقييمًا إيجابيًّا.

الخطوة التالية هي إسناد أعمدة النص والقيم إلى متغيرات مستقلة في أمثلة التدريب والاختبار المُمثّلة كمجموعة بيانات DataFrame كما يلي:

```python
# extract the text from the 'text' column for both training and testing.
X_train_text=imdb_train_reviews['text']
X_test_text=imdb_test_reviews['text']

# extract the labels from the 'label' column for both training and testing.
Y_train=imdb_train_reviews['label']
Y_test=imdb_test_reviews['label']
X_train_text  # training data in text format
```

يُستخدم الرمزان X وY عادةً في التعلُّم الموجَّه فيعبّر X عن البيانات المدخلة للتنبؤ، وY عن القيم المستهدفة.

```
0        I grew up (b. 1965) watching and loving the Th...
1        When I put this movie in my DVD player, and sa...
2        Why do people who do not know what a particula...
3        Even though I have great interest in Biblical ...
4        Im a die hard Dads Army fan and nothing will e...
                               ...
39995    "Western Union" is something of a forgotten cl...
39996    This movie is an incredible piece of work. It ...
39997    My wife and I watched this movie because we pl...
39998    When I first watched Flatliners, I was amazed....
39999    Why would this film be so good, but only gross...
Name: text, Length: 40000, dtype: object
```

شكل 3.4: صورة من أمثلة التدريب (X__train__text) من مجموعة بيانات DataFrame

وزارة التعليم
Ministry of Education
2025 - 1447

## تجهيز البيانات والمعالجة المُسبقة Data Preparation and Pre-Processing

على الرغم من أن تنسيق النص الأولي كما في الشكل 3.4 بديهي للقارئ البشري، إلا أنَّ خوارزميات التعلُّم الموجَّه لا تستطيع التعامل معه بصورته الحالية. فبدلًا من ذلك، تحتاج الخوارزميات إلى تحويل هذه المُستنَدات إلى تنسيق **متَّجَه رقمي** (Numeric Vector). فيما يُعرف بعملية **البرمجة الاتجاهية** (Vectorization). ويمكن تطبيق عملية البرمجة الاتجاهية بعدة طرائق مختلفة، وتتميز بأن لها تأثيرًا إيجابيًّا كبيرًا على أداء النموذج المُدرّب.

### مكتبة سكليرن Sklearn Library

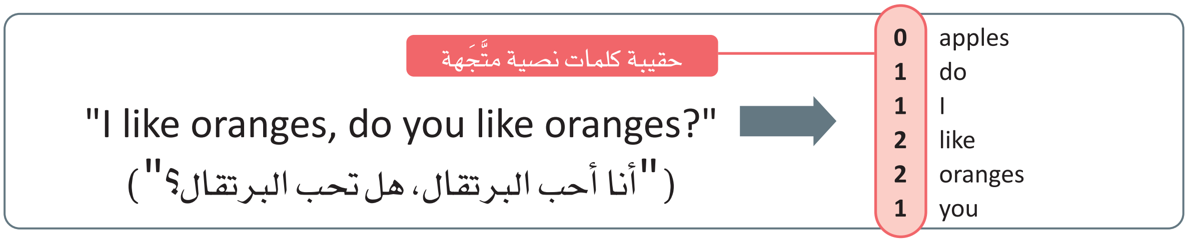
سيتم بناء النموذج الموجَّه باستخدام مكتبة سكليرن وتُعرف كذلك باسم مكتبة **سايكيت ليرن** (Scikit-Learn)، وهي مكتبة شهيرة في البايثون تختص بتعلُّم الآلة. توفر المكتبة مجموعة من الأدوات والخوارزميات لأداء مهام متعددة، مثل: التصنيف، والانحدار، والتجميع، وتقليص الأبعاد. إحدى الأدوات المفيدة في مكتبة سكليرن هي أداة تُسمى CountVectorizer، ويمكن استخدامها في تهيئة عملية المعالجة وتمثيل البيانات النصية بالمتَّجَهات.

### أداة CountVectorizer

تُستخدم أداة CountVectorizer في تحويل مجموعة من المُستنَدات النصية إلى مصفوفة من رموز متعددة، حيث يمثّل كلّ صفّ مستندًا وكل عمود يمثل رمزًا خاصًا. قد تكون الرموز كلمات فردية أو عبارات أو بُنيات أكثر تعقيدًا تقوم بالتقاط الأنماط المتعددة من البيانات النصية الأساسية. تُشير المُدخَلات في المصفوفة إلى عدد مرات ظهور الرمز في كل مستند. ويُعرف ذلك أيضًا باسم تمثيل **حقيبة الكلمات** (BoW) "bag-of-words"، حيث يتجاهل ترتيب الكلمات في النص مع المحافظة على تكرارها فيه. على الرغم من أن تمثيل حقيبة الكلمات هو تبسيط شديد للغة البشرية، إلا أنه يحقق نتائج تنافسية للغاية عند التطبيق العملي.

**البرمجة الاتجاهية (Vectorization):**

البرمجة الاتجاهية هي عملية تحويل السلاسل النصية المكوَّنة من الكلمات أو العبارات (النص) إلى متَّجَه متجانس من الأرقام الحقيقية يُستخدم لترميز خصائص النص باستخدام تنسيق تفهمه خوارزميات تعلُّم الآلة.




شكل 3.5: تمثيل حقيبة الكلمات (bag-of-words)

يستخدِم المقطع البرمجي التالي أداة CountVectorizer لتمثيل مجموعة بيانات التدريب IMDb بالمتَّجَهات:

```python
from sklearn.feature_extraction.text import CountVectorizer

# the min_df parameter is used to ignore terms that appear in less than 10 reviews.
vectorizer_v1 = CountVectorizer(min_df=10)

vectorizer_v1.fit(X_train_text) # fit the vectorizer on the training data.
# use the fitted vectorizer to vectorize the data.
X_train_v1 = vectorizer_v1.transform(X_train_text)

X_train_v1
```

```
<40000x23392 sparse matrix of type '<class 'numpy.int64'>'
	with 5301561 stored elements in Compressed Sparse Row format>
```

```python
# expand the sparse data into a sparse matrix format, where each column represents a different word.
X_train_v1_dense=pd.DataFrame(X_train_v1.toarray(),
                  columns=vectorizer_v1.get_feature_names_out())
X_train_v1_dense
```

| | 00 | 000 | 007 | 01 | 02 | 04 | 05 | 06 | 07 | 08 | ... | zoo | zoom | zooming | zooms | zorro | zu | zucco | zucker | zulu | über |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **2** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **3** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **4** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **39995** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **39996** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **39997** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **39998** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **39999** | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

40000 rows × 23392 columns

شكل 3.6: تمثيل مجموعة بيانات التدريب بالمتَّجَهات

يُعبِّر هذا التنسيق **الكثيف** (Dense) للمصفوفة عن 40,000 تقييم ومراجعة في بيانات التدريب. تحتوي المصفوفة على عمود لكل كلمة تظهر في 10 مراجعات على الأقل (مُنفذة بواسطة المتغير min_df). كما يتضح بالأعلى، ينتج عن ذلك 23,392 عمودًا، مرتبة في ترتيب أبجدي رقمي. يُعبِّر مُدخَل المصفوفة في الموضع [i,j] عن عدد المرات التي تظهر فيها كلمة j في المراجعة i. وعلى الرغم من إمكانيّة استخدام هذه المصفوفة مباشرةً من قِبَل خوارزمية التعلُّم الموجَّه، إلا أنها غير فعّالة من حيث استخدام الذاكرة. والسبب في ذلك أن الغالبية العظمى من المُدخَلات في هذه المصفوفة تساوي 0. وهذا يحدث لأن نسبة ضئيلة جدًا فقط من بين 23,392 كلمة محتملة ستظهر فعليًا في كل مراجعة. ولمعالجة هذا القصور، تُخزِّن أداة CountVectorizer البيانات الممثلة بالمتَّجَهات في مصفوفة متباعدة، حيث تحتفظ فقط بالمُدخَلات غير الصفرية في كل عمود. يستخدِم المقطع البرمجي بالأسفل الدالة getsizeof() التي تحدِّد حجم الكائنات في لغة **البايثون** (Python) **بالبايتات** (Bytes) لتوضيح مدى التوفير في الذاكرة عند استخدام المصفوفة المتباعدة لبيانات IMDb:

```python
from sys import getsizeof
print('\nMegaBytes of RAM memory used by the raw text format:',
         getsizeof(X_train_text)/1000000)
print('\nMegaBytes of RAM memory used by the dense matrix format:',
         getsizeof(X_train_v1_dense)/1000000)
print('\nMegaBytes of RAM memory used by the sparse format:',
         getsizeof(X_train_v1)/1000000)
```

```
MegaBytes of RAM memory used by the raw text format: 54.864133

MegaBytes of RAM memory used by the dense matrix format: 7485.440144

MegaBytes of RAM memory used by the sparse format: 4.8e-05
```

وبحسب المتوقّع تحتاج المصفوفة المتباعدة إلى ذاكرة أقل بكثير وتحديدًا 0.000048 ميجابايت، بينما تشغل المصفوفة الكثيفة 7 جيجابايت، كما أنّ هذه المصفوفة لن تُستخدَم مرة أخرى وبالتالي يمكن حذفها لتوفير هذا الحجم الكبير من الذاكرة:

```
# delete the dense matrix.
del X_train_v1_dense
```

# بِناء خط أنابيب التنبؤ

## Building a Prediction Pipeline

الآن بعد أن تمكّنت من تمثيل بيانات التدريب بالمتّجهات فإن الخطوة التالية هي بِناء خط أنابيب التنبؤ الأول. وللقيام بذلك، ستستخدِم نوعًا من المُصنِّفات يسمى **مُصنِّف بايز الساذج** (Naive Bayes Classifier)، حيث يَستخدِم هذا المصنِّف احتمالات الكلمات أو العبارات المحددة الواردة في النَّص للتنبؤ باحتمال انتمائه إلى تصنيف محدد. جاءت كلمة **الساذج** (Naive) في اسم المُصنِّف من افتراض أن وجود كلمة بعينها في النَّص مستقل عن وجود أي كلمة أخرى. وهذا افتراض قوي، ولكنه يسمح بتدريب الخوارزمية بسرعة وبفعالية كبيرة.

**المُصنِّف (Classifier):**

المُصنِّف في تعلُّم الآلة هو نموذج يُستخدم لتمييز نقاط البيانات في فئات أو تصنيفات مختلفة. الهدف من المُصنِّف هو التعلُّم من بيانات التدريب المُعنوَنة، ومن ثَمَّ القيام بالتنبؤات حول قيم التصنيف لبيانات جديدة.

يستخدِم المقطع البرمجي التالي تطبيق **مُصنِّف بايز الساذج** (Multinomial NB) من **مكتبة سكليرن** (Sklearn Library) لتدريب نموذج التعلُّم الموجَّه على بيانات التدريب IMDb بالمتَّجهات:

```
from sklearn.naive_bayes import MultinomialNB

model_v1=MultinomialNB() # a Naive Bayes Classifier

model_v1.fit(X_train_v1, Y_train) # fit the classifier on the vectorized training data.

from sklearn.pipeline import make_pipeline

# create a prediction pipeline: first vectorize using vectorizer_v1, then use model_v1 to predict.
prediction_pipeline_v1 = make_pipeline(vectorizer_v1, model_v1)
```

على سبيل المثال، سيُنتج هذا المقطع البرمجي مصفوفة نتائج يرمز فيها الرقم 1 للتقييم الإيجابي و0 للتقييم السلبي:

```
prediction_pipeline_v1.predict(['One of the best movies of the year. Excellent
                                 cast and very interesting plot.',
                                'I was very disappointed with his film. I
                                 lost all interest after 30 minutes' ])
```

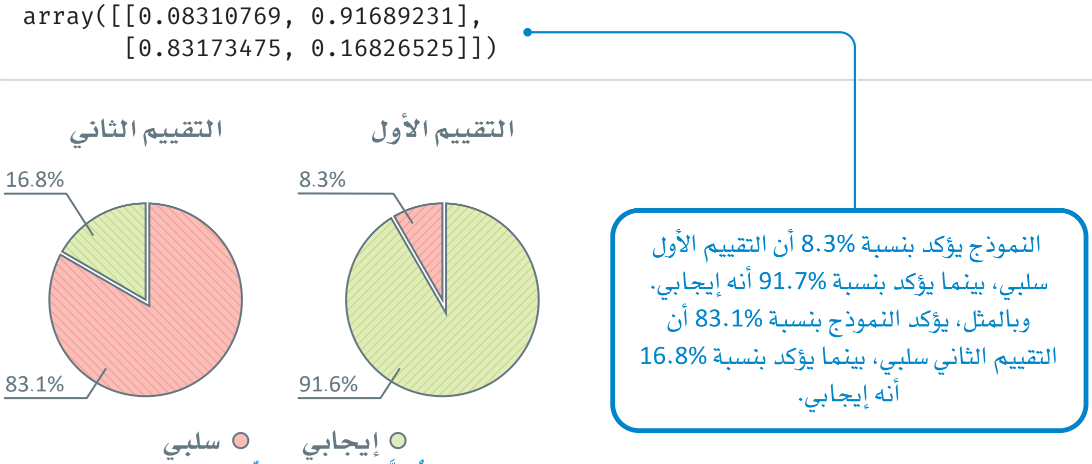
```
array([1, 0], dtype=int64)
```


وزارة التعليم
Ministry of Education
2025 - 1447

يتنبأ خط الأنابيب بشكل صحيح بالقيمة الإيجابية للتقييم الأول والقيمة السلبية للتقييم الثاني. يُمكن استخدام الدالة المُضمّنة ()predict_proba لتحديد جميع الاحتمالات التي يقوم خط الأنابيب بتخصيصها لكل واحدة من القيمتين المحتملتين. العنصر الأول هو احتمال تعيين 0 والعنصر الثاني هو احتمال تعيين 1:

```
prediction_pipeline_v1.predict_proba(['One of the best movies of the year. Ex
                                      cellent cast and very interesting plot.',
                                      'I was very disappointed with his film.
                                       I lost all interest after 30 minutes' ])
```

```
array([[0.08310769, 0.91689231],
       [0.83173475, 0.16826525]])
```




شكل 3.7: مُخطَّطان دائريان يوضِّحان النسب المئوية للتقييمين

الخطوة التالية هي اختبار دقة خط الأنابيب الجديد في تصنيف التقييمات في مجموعة بيانات اختبار IMDb. المُخرَج هو مصفوفة تشمل جميع قيم نتائج تصنيف التقييمات الواردة في بيانات الاختبار:

```
# use the pipeline to predict the labels of the testing data.
predictions_v1 = prediction_pipeline_v1.predict(X_test_text) # vectorize the text
data, then predict.

predictions_v1
```

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

توفر لغة البايثون العديد من الأدوات لتحليل وتصوير نتائج خطوط أنابيب التصنيف. تشمل الأمثلة دالة ()accuracy_score من مكتبة سكليرن وتمثيل **مصفوفة الدقة** (Confusion Matrix) من مكتبة **سايكيت بلوت** (Scikit-Plot)، وهناك مقاييس تقييم أخرى مثل: الدقة، والاستدعاء، والنوعية، والحساسية، ومقياس درجة F1، وفقًا لحالة الاستخدام التي يمكن حسابها من مصفوفة الدقة. المُخرَج التالي هو تقريب دقيق لدرجة التنبؤ:

```
from sklearn.metrics import accuracy_score
accuracy_score(Y_test, predictions_v1) # get the achieved accuracy.
```

```
0.8468
```

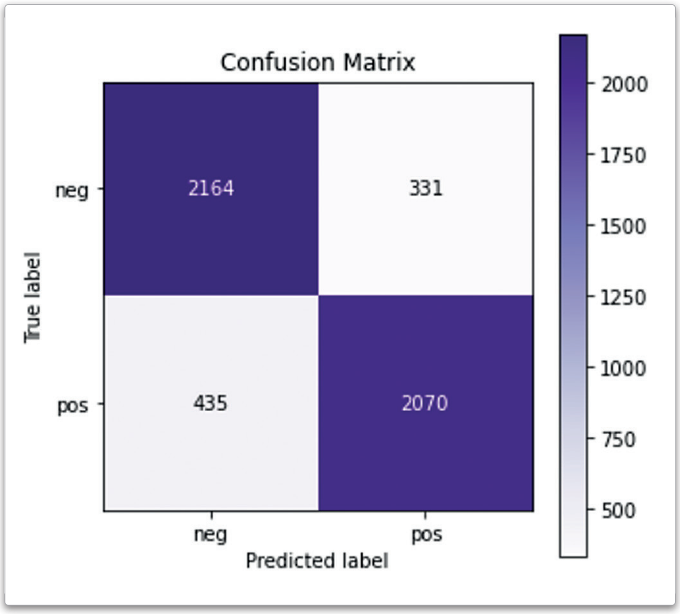
وزارة التعليم
Ministry of Education
2025 - 1447

```
%%capture
!pip install scikit-plot; # install the scikit-plot library, if it is missing.
import scikitplot; # import the library

class_names=['neg','pos'] # pick intuitive names for the 0 and 1 labels.

# plot the confusion matrix.
scikitplot.metrics.plot_confusion_matrix(
                                    [class_names[i] for i in Y_test],
                                    [class_names[i] for i in predictions_v1],
                                     title="Confusion Matrix", # title to use
                                     cmap="Purples", # color palette to use
                                     figsize=(5,5) # figure size
                                     );
```

القيم الحقيقية.

القيم المُتوقَّعة.

تحتوي مصفوفة الدقة على عدد التصنيفات الحقيقية مقابل المُتوقَّعة. في مُهِمَّة التصنيف الثنائية (مثل: مسألة احتواء قيمتين، الموجودة في مُهِمَّة IMDb)، ستحتوي مصفوفة الدقة على أربع خلايا:




**التنبؤات السالبة الصحيحة (أعلى اليسار):**
عدد المرات التي تنبأ فيها المُصنِّف بالحالات السالبة بشكل صحيح.

**التنبؤات السالبة الخاطئة (أعلى اليمين):**
عدد المرات التي تنبأ فيها المُصنِّف بالحالات السالبة بشكل خاطئ.

**التنبؤات الموجبة الخاطئة (أسفل اليسار):**
عدد المرات التي تنبأ فيها المُصنِّف بالحالات الموجبة بشكل خاطئ.

**التنبؤات الموجبة الصحيحة (أسفل اليمين):**
عدد المرات التي تنبأ فيها المُصنِّف بالحالات الموجبة بشكل صحيح.

شكل 3.8: نتائج مصفوفة الدقة بتطبيق مصنِّف بايز الساذج على بيانات الاختبار باستخدام مجموعة بيانات IMDb

تُظهر النتائج أنه على الرغم من أن خط الأنابيب الأول يحقق دقة تنافسية تصل إلى 84.68%، إلا أنه لا يزال يُخطئ في تصنيف مئات التقييمات. فهناك 331 تنبّؤًا غير صحيح في الربع الأيمن العلوي و435 تنبّؤًا غير صحيح في الربع الأيسر السفلي، بإجمالي 766 تنبّؤًا غير صحيح. الخطوة الأولى نحو تحسين الأداء هي دراسة سلوك خط أنابيب التنبؤ، لمعرفة كيف يقوم بمعالجة النصّ وفهمه.

**الدقة (Accuracy):**

الدقة هي نسبة التنبؤات الصحيحة إلى إجمالي عدد التنبؤات.

$$\text{الدقة} = \frac{(\text{التنبؤات الموجبة الصحيحة} + \text{التنبؤات السالبة الصحيحة})}{(\text{التنبؤات الموجبة الصحيحة} + \text{التنبؤات السالبة الصحيحة} + \text{التنبؤات الموجبة الخاطئة} + \text{التنبؤات السالبة الخاطئة})}$$


وزارة التعليم
Ministry of Education
2025 - 1447

## شرح مُتنبِّئات الصندوق الأسود Explaining Black-Box Predictors

يستخدِم مصنَّف بايز الساذج الصيغ الرياضية البسيطة لتجميع احتمالات آلاف الكلمات وتقديم تنبؤاتها. وبالرغم من بساطة النموذج، إلا أنه لا يزال غير قادر على تقديم شرح بسيط ومباشر لكيفية قيام النموذج بتوقُّع القيمة الموجبة أو السالبة لجزء محدد من النص. قارِن ذلك مع مُصنِّفات شجرة القرار الأكثر وضوحًا، حيث يتم تمثيل القواعد التي تعلمها النموذج في الهيكل الشجري، مما يُسهِّل على الأشخاص فهم كيف يقوم المُصنِّف بالتنبؤات. يتيح هيكل الشجرة كذلك الحصول على تصور مرئي للقرارات المُتخذَّة في كل فرع، ممّا يكون مفيدًا في فهم العلاقات بين الخصائص المُدخَلة والمتغير المستهدف.

الافتقار إلى قدرة التفسير تمثل تحديًا كبيرًا في الخوارزميات الأكثر تعقيدًا، كتلك المُستنِدة إلى التجميعات مثل: توليفات من الخوارزميات المتعددة أو الشبكات العصبية. فبدون القدرة على التفسير، تتقلص خوارزميات التعلُّم الموجَّه إلى متنبئات الصندوق الأسود: على الرغم من أنها تفهم النص بشكل كافٍ للتنبؤ بالقيم، إلا أنها لا تزال غير قادرة على تفسير كيف تقوم باتخاذ القرار. أجريت العديد من الأبحاث للتغلب على هذه التحديات بتصميم وسائل قادرة على التفسير تستطيع فهم نماذج الصندوق الأسود. وواحدة من الوسائل الأكثر شهرة هي **النموذج المحايد المحلي القابل للتفسير والشرح** (Local Interpretable Model-Agnostic Explanations – LIME).

### النموذج المحايد المحلي القابل للتفسير والشرح
### Local Interpretable Model-Agnostic Explanations - LIME

**النموذج المحايد المحلي القابل للتفسير والشرح** (LIME) هو طريقة لتفسير التنبؤات التي قامت بها نماذج الصندوق الأسود. وذلك من خلال النظر في نقطة بيانات واحدة في وقت محدد، وإجراء تغييرات بسيطة عليها لمعرفة كيف يؤثر ذلك على قدرة تنبؤ النموذج، ثم تُستخدم هذه المعلومات لتدريب نموذج مفهوم وبسيط مثل الانحدار الخطي على تفسير هذه التنبؤات. بالنسبة للبيانات النصية، يقوم النموذج المحايد المحلي القابل للتفسير والشرح بالتعرّف على الكلمات أو العبارات التي لها الأثر الأكبر على القيام بالتنبؤات.

وفيما يلي، تطبيق بلغة البايثون يوضِّح ذلك:

```
%%capture

!pip install lime  # install the lime library, if it is missing
from lime.lime_text import LimeTextExplainer

# create a local explainer for explaining individual predictions
explainer_v1 = LimeTextExplainer(class_names=class_names)

# an example of an obviously negative review
easy_example='This movie was horrible. The actors were terrible and the plot 
was very boring.'

# use the prediction pipeline to get the prediction probabilities for this example
print(prediction_pipeline_v1.predict_proba([easy_example]))
```

```
[[0.99874831 0.00125169]]
```

كما هو مُتوقَّع، يقدم نموذج التنبؤ تنبؤًا سلبيًا مؤكدًا بدرجة كبيرة في هذا المثال البسيط.

```
# explain the prediction for this example.
exp = explainer_v1.explain_instance(easy_example.lower(),
                                    prediction_pipeline_v1.predict_proba,
                                    num_features=10)
# print the words with the strongest influence on the prediction.
exp.as_list()
```

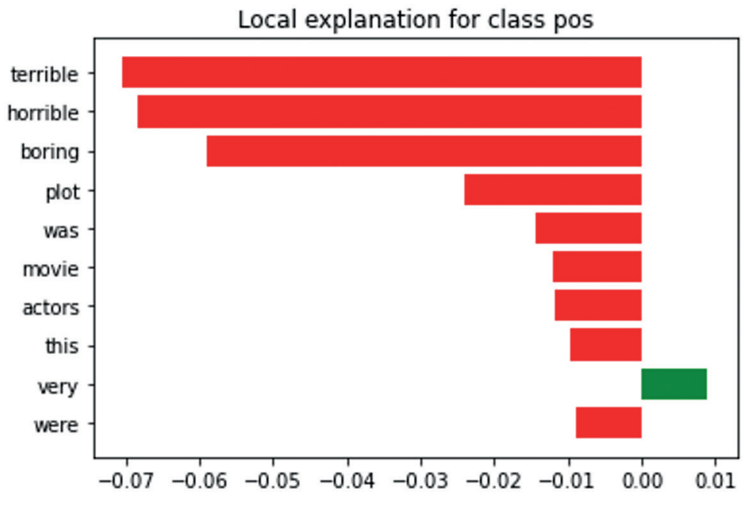
```
[('terrible', -0.07046118794796816),
 ('horrible', -0.06841672591649835),
 ('boring', -0.05909016205135171),
 ('plot', -0.024063095577996376),
 ('was', -0.014436071624747861),
 ('movie', -0.011956911011210977),
 ('actors', -0.011682594571408675),
 ('this', -0.009712387273986628),
 ('very', 0.008956707731803237),
 ('were', -0.008897098392433257)]
```

الخصائص العشرة الأكثر تأثيرًا.

الدرجة المقابلة لكل كلمة تمثل مُعاملًا في نموذج الانحدار الخطي البسيط المُستخدَم لتقديم التفسير.

يمكن الحصول على تصور مرئي أكثر دقةً على النحو التالي:

```
# visualize the impact of the most influential words.
fig = exp.as_pyplot_figure()
```



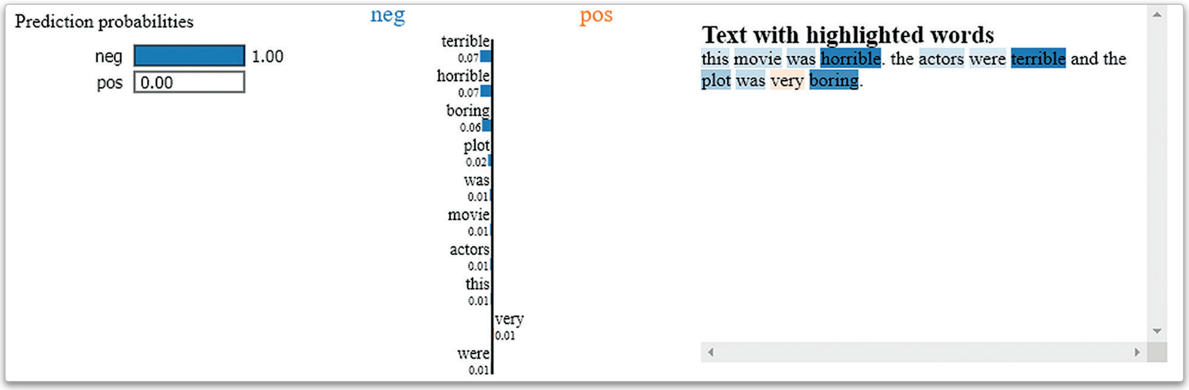

شكل 3.9: الكلمات الأعلى تأثيرًا في القيام بالتنبؤات


وزارة التعليم
Ministry of Education
2025 - 1447

يَزيد المُعامِل السالب من احتمالية التصنيف السالب، بينما يُقلل المُعامِل الموجب منه. على سبيل المثال، الكلمات: horrible (فظيع)، وterrible (مريع)، وboring (ممل) لها التأثير الأقوى على قرار النموذج بالتنبؤ بالقيمة السالبة. الكلمة very (جدًا) دفعت النموذج قليلًا في اتجاه آخر إيجابي، ولكنها لم تكن كافية لتغيير القرار. بالنسبة للمراقب البشري، قد يبدو غريبًا أن الكلمات الخالية من المشاعر مثل: plot (الحبكة الدرامية) أو was (كان) لها مُعامِلات مرتفعة نسبيًا. ومع ذلك، من الضروري أن تتذكر أن تعلُّم الآلة لا يتبع دومًا الوعي البشري السليم.

وقد تكشف هذه المُعامِلات المرتفعة بالفعل عن قصور في منطق الخوارزمية وقد تكون مسؤولة عن بعض أخطاء نموذج التنبُّؤ. وعلى نحوٍ بديل، يُعدُّ نموذج التنبُّؤ بمثابة مؤشرٍ على الأنماط التنبؤية الكامنة والغنيّة في الوقت نفسه بالمعلومات. على سبيل المثال، قد يبدو الواقع وكأن المُقيِّمين البشريين أكثر استخدامًا لكلمة plot (الحبكة الدرامية) أو صيغة الماضي was (كان) عند الحديث في سياق سلبي. ويمكن لمكتبة **النموذج المحايد المحلي القابل للتفسير والشرح** (LIME) في لغة البايثون تصوير الشروحات بطرائق أخرى. على سبيل المثال:

```
exp.show_in_notebook()
```




شكل 3.10: التمثيلات المرئية الأخرى

التقييم المُستخدَم في المثال السابق كان سلبيًا بشكل واضح ويَسهُل التنبؤ به. خُذْ بعين الاعتبار التقييم التالي الأكثر صعوبةً والذي يمكن أن يتسبب في تذبذب دقة الخوارزمية، وهو مأخوذ من مجموعة بيانات اختبار IMDb:

```
# an example of a positive review that is mis-classified as negative by prediction_pipeline_v1
mistake_example= X_test_text[4600]
mistake_example
```

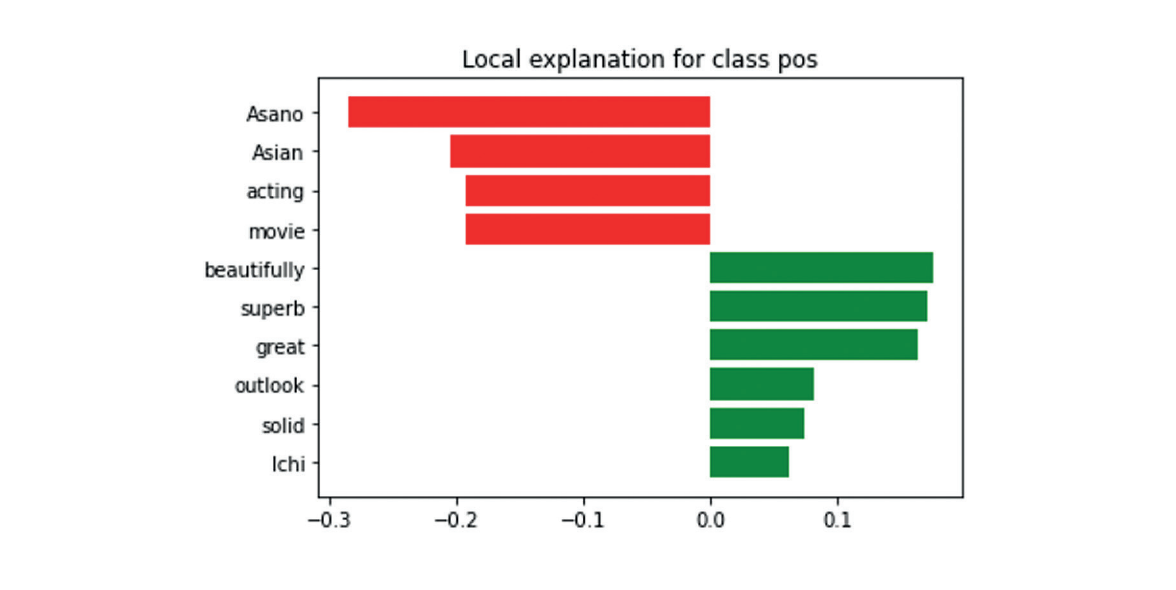
```
"I personally thought the movie was pretty good, very good acting by
Tadanobu Asano of Ichi the Killer fame. I really can't say much about the
story, but there were parts that confused me a little too much, and overall
I thought the movie was just too lengthy. Other than that however, the
movie contained superb acting great fighting and a lot of the locations
were beautifully shot, great effects, and a lot of sword play. Another
solid effort by Tadanobu Asano in my opinion. Well I really can't say
anymore about the movie, but if you're only outlook on Asian cinema is
Crouching Tiger Hidden Dragon or House of Flying Daggers, I would suggest
you trying to rent it, but if you're a die-hard Asian cinema fan I would
say this has to be in your collection very good Japanese film."
```

```
# get the correct labels of this example.
print('Correct Label:', class_names[Y_test[4600]])

# get the prediction probabilities for this example.
print('Prediction Probabilities for neg, pos:',
      prediction_pipeline_v1.predict_proba([mistake_example]))
```

```
Correct Label: pos
Prediction Probabilities for neg, pos: [[0.8367931 0.1632069]]
```

على الرغم من أن هذا التقييم إيجابي بشكل واضح، إلا أنّ نموذج التنبُّؤ قدّم تنبؤًا سلبيًا مؤكدًا للغاية باحتمالية وصلت إلى 83%. يمكن الآن استخدام المُفسِّر لتوضيح السبب وراء اتخاذ نموذج التنبُّؤ مثل هذا القرار الخاطئ:

```
# explain the prediction for this example.
exp = explainer_v1.explain_instance(mistake_example, prediction_pipeline_
v1.predict_proba, num_features=10)

# visualize the explanation.
fig = exp.as_pyplot_figure()
```




شكل 3.11: الكلمات التي أثرت على القرار الخاطئ

على الرغم من أن نموذج التنبُّؤ يستنبط التأثير الإيجابي لبعض الكلمات على نحوٍ صحيح مثل:

beautifully (بشكل جميل)، وgreat (رائع)، وsuperb (مدهش)، إلا أنّه يتّخذ في النهاية قرارًا سلبيًا استنادًا إلى العديد من الكلمات التي يبدو أنها لا تعبِّر بشكل واضح عن المشاعر السلبية مثل: Asano (أسانو)، وAsian (آسيوي)، وmovie (فيلم)، وacting (تمثيل).

وهذا يوضِّح العيوب الكبيرة في المنطِق الذي يستخدمه نموذج التنبُّؤ لتصنيف المفردات الواردة في نصوص التقييمات المُقدمة. القسم التالي يوضِّح كيف أن تحسين هذا المنطق يمكن أن يطور من أداء نموذج التنبُّؤ إلى حدٍ كبير.


وزارة التعليم
Ministry of Education
2025 - 1447

# تحسين البرمجة الاتجاهية للنصوص

## Improving Text Vectorization

استخدم الإصدار الأول لخط أنابيب التنبؤ أداة CountVectorizer لحساب عدد المرات التي تظهر فيها كل كلمة في كل تقييم. تتجاهل هذه المنهجية حقيقتين أساسيتين حول اللغات البشرية:

- قد يتغير معنى الكلمة وأهميّتها حسب الكلمات المُستخدَمة معها.
- تكرار الكلمة في المُستنَد لا يُعدُّ دومًا تمثيلًا دقيقًا لأهميّتها. على سبيل المثال، على الرغم من أن تكرار كلمة great (رائع) مرتين قد يمثل مؤشرًا إيجابيًا في مستند يحتوي على 100 كلمة، إلا أنه يمثل مؤشرًا أقل أهمية بكثير في مستند يحتوي على 1000 كلمة.

**التعبير النمطي (Regular Expression):**

التعبير النمطي هو نمط نص يُستخدَم لمطابقة ولمعالجة سلاسل النصوص وتقديم طريقة موجزة ومرنة لتحديد أنماط النصوص، كما تُستخدَم على نطاق واسع في معالجة النصوص وتحليل البيانات.

سيشرح هذا الجزء كيفية تحسين البرمجة الاتجاهية للنصوص لأخذ هاتين الحقيقتين في عين الاعتبار. يستدعي المقطع البرمجي التالي ثلاثة مكتبات مختلفة بلغة البايثون، ستُستخدم لتحقيق ذلك:

- **nltk** وجينسم (Gensim): تُستخدَم هاتان المكتبتان الشّهيرتان في مهام معالجة اللغات الطبيعية المُتنوّعة.
- **re**: تُستخدم هذه المكتبة في البحث عن النّصوص، ومعالجتها باستخدام التعبيرات النمطية.

```
%%capture

!pip install nltk # install nltk
!pip install gensim # install gensim

import nltk # import nltk
nltk.download('punkt') # install nltk's tokenization tool, used to split a text into sentences.

import re  # import re

from gensim.models.phrases import Phrases, ENGLISH_CONNECTOR_WORDS # import tools
from the gensim library.
```

### تحديد العبارات Detecting Phrases

يمكن استخدام الدالة الآتية لتقسيم مستند محدد إلى قائمة من الجُمل المُقسَّمة، حيث يمكن تمثيل كل جملة مُقسَّمة بقائمة من الكلمات:

**التقسيم (Tokenization):**

يقصد به: عملية تقسيم البيانات النصية إلى أجزاء مثل كلمات، وجُمل، ورموز، وعناصر أخرى يطلق عليها **الرموز** (Tokens).

```
# convert a given doc to a list of tokenized sentences.
def tokenize_doc(doc:str):
    return [re.findall(r'\b\w+\b',
            sent.lower()) for sent in nltk.sent_tokenize(doc)]
```

دالة ( )sent_tokenize تُقسِّم المُستنَد إلى قائمة من الجُمل.

دالة ( )sent_tokenize من مكتبة nltk تُقسِّم المُستنَد إلى قائمة من الجُمل.

بعد ذلك، يتم كتابة كل جملة بأحرف صغيرة وتغذيتها إلى دالة ( )findall من مكتبة re لتقوم بتحديد تكرارات التعبيرات النمطية '\b\w+\b'. ستختبرها على السلسلة النصية الموجودة في متغير raw_text في هذا السياق:

- ‎\w تتطابق مع كل الرموز الأبجدية الرقمية (a-z ،A-Z ،0-9) والشَرطة السفلية.
- ‎\w+ تُستَخدم للبحث عن واحد أو أكثر من رموز w\. لذلك، في السلسلة النصية hello123_world (مرحبًا 123_ العالم)، النمط w+\ سيتطابق مع الكلمات hello (مرحبًا) و 123 وworld (العالم).
- ‎\b تمثل **الفاصل** (Boundry) بين رمز w\ ورمز ليس w\، وكذلك في بداية أو نهاية السلسلة النصية المُعطاة. على سبيل المثال: سيتطابق النمط bcat\b\ مع الكلمة cat (القطة) في السلسلة النصية The cat is cute (القطة لطيفة)، ولكنه لن يتطابق مع الكلمة cat (القطة) في السلسلة النصية The category is pets (فئة الحيوانات الأليفة).

أدناه مثالًا على التقسيم باستخدام الدالة tokenize_doc().

```
raw_text='The movie was too long. I fell asleep after the first 2 hours.'
tokenized_sentences=tokenize_doc(raw_text)
tokenized_sentences
```

```
[['the', 'movie', 'was', 'too', 'long'],
 ['i', 'fell', 'asleep', 'after', 'the', 'first', '2', 'hours']]
```

يمكن الآن تجميع الدالة tokenize_doc() مع أداة العبارات من مكتبة **جينسم** (Gensim) لإنشاء نموذج العبارة، وهو نموذج يمكنه التعرّف على العبارات المكونة من عدة كلمات في جملة معطاة. يستخدِم المقطع البرمجي التالي بيانات التدريب IMDB الخاصة بـ (X_train_text) لبناء مثل هذا النموذج:

```
sentences=[] # list of all the tokenized sentences across all the docs in this dataset

for doc in X_train_text: # for each doc in this dataset
    sentences+=tokenize_doc(doc) # get the list of tokenized sentences in this doc

# build  a phrase model on the given data
imdb_phrase_model = Phrases(sentences, ❶
                            connector_words=ENGLISH_CONNECTOR_WORDS, ❷
                            scoring='npmi', ❸
                            threshold=0.25).freeze() ❹
```

كما هو موضَّح بالأعلى، تَستقبل الدالة Phrases() أربعة متغيرات:

❶ قائمة الجُمل المُقسَّمة من مجموعة النصوص المُعطاة.

❷ قائمة بالكلمات الإنجليزية الشائعة التي تظهر بصورة متكررة في العبارات (مثل: the، و of)، وليس لها أي قيمة موجبة أو سالبة، ولكن يمكنها إضفاء المشاعر حسب السياق، ولذلك يتم التعامل معها بصورة مختلفة.

❸ تُستَخدم دالة تسجيل النقاط لتحديد ما إذا كان تضمين مجموعة من الكلمات في العبارة نفسها واجبًا. المقطع البرمجي بالأعلى يَستخدِم مقياس **المعلومات النقطية المشتركة المُعايَر** (Normalized Pointwise Mutual Information – NPMI) لهذا الغرض. يستند هذا المقياس على تكرار توارد الكلمات في العبارة المُرشحة وتكون قيمته بين 1− ويرمز إلى **الاستقلالية الكاملة** (Complete Independence)، و 1+ ويرمز إلى **التوارد الكامل** (Complete Co-occurrence).

❹ في حدود دالة تسجيل النقاط يتم تجاهل العبارات ذات النقاط الأقل. ومن الناحية العملية، يمكن ضبط هذه الحدود لتحديد القيمة التي تُعطي أفضل النتائج في التطبيقات النهائية مثل: النمذجة التنبؤية.

تُحوِّل دالة freeze() نموذج العبارة إلى تنسيق غير قابل للتغيير أي **مُجمَّد** (Frozen) لكنّه أكثر سرعة.


وزارة التعليم
Ministry of Education
2025 - 1447

عند تطبيقها على الجملتين المُقسَّمتين بالمثال المُوضح بالأعلى، سيُحقق نموذج العبارة النتائج التالية:

```
imdb_phrase_model[tokenized_sentences[0]]
```

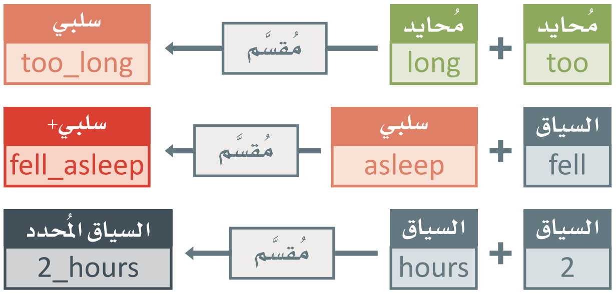
```
['the', 'movie', 'was', 'too_long']
```

```
imdb_phrase_model[tokenized_sentences[1]]
```

```
['i', 'fell_asleep', 'after', 'the', 'first', '2_hours']
```

يحدِّد نموذج العبارة ثلاثة عبارات على النحو التالي: fell_asleep (سقط نائمًا) وtoo_long (طويل جدًا)، و2_hours (2-ساعة) وجميعها تحمل معلومات أكثر من كلماتها المفردة.

على سبيل المثال، تحمل عبارة too_long (طويل جدًا) مشاعر سلبية واضحة، على الرغم من أن كلمتي too (جدًا) وlong (طويل) لا تعبران عن ذلك منفردتين، وبالمثل، فعلى الرغم من أن كلمة asleep (نائم) في مراجعة الفيلم تمثل دلالة سلبية، فالعبارة fell_asleep (سقط نائمًا) توصل رسالة أكثر وضوحًا. وأخيرًا، تستنبط من 2_hours (2-ساعة) سياقًا أكثر تحديدًا من الكلمتين 2 وhours كلٍّ على حدة.




شكل 3.12: المشاعر الإيجابية والسلبية قبل التقسيم وبعده

تستخدِم الدالة التالية إمكانية تحديد العبارات بهذا الشكل لتفسير العبارات في وثيقة مُعطاه:

```
def annotate_phrases(doc:str, phrase_model):

    sentences=tokenize_doc(doc)# split the document into tokenized sentences.

    tokens=[] # list of all the words and phrases found in the doc
    for sentence in sentences: # for each sentence
        # use the phrase model to get tokens and append them to the list.
        tokens+=phrase_model[sentence]
    return ' '.join(tokens) # join all the tokens together to create a new annotated document.
```

يستخدِم المقطع البرمجي التالي دالة annotate_phrases() لتفسير كلٍ من تقييمات التدريب والاختبار من مجموعة بيانات IMDb.

```
# annotate all the test and train reviews.
X_train_text_annotated=[annotate_phrases(doc,imdb_phrase_model) for doc in X_
train_text]
X_test_text_annotated=[annotate_phrases(text,imdb_phrase_model)for text in X_
test_text]
```

وزارة التعليم
Ministry of Education
2025 - 1447

```
# an example of an annotated document from the imdb training data
X_train_text_annotated[0]
```

```
'i_grew up b 1965 watching and loving the thunderbirds all my_mates at school watched
we played thunderbirds before school during lunch and after school we all wanted to
be virgil or scott no_one wanted to be alan counting down from 5 became an art_form
i took my children to see the movie hoping they would get_a_glimpse of what i_loved
as a child how bitterly disappointing the only high_point was the snappy theme_tune
not that it could compare with the original score of the thunderbirds thankfully
early saturday_mornings one television_channel still plays reruns of the series
gerry_anderson and his_wife created jonatha frakes should hand in his directors chair
his version was completely hopeless a waste of film utter_rubbish a cgi remake may_be
acceptable but replacing marionettes with homo_sapiens subsp sapiens was a huge error
of judgment'
```

## استخدام مقياس تكرار المصطلح-تكرار المُستنَد العكسي في البرمجة الاتجاهية للنصوص

## Using TF-IDF for Text Vectorization

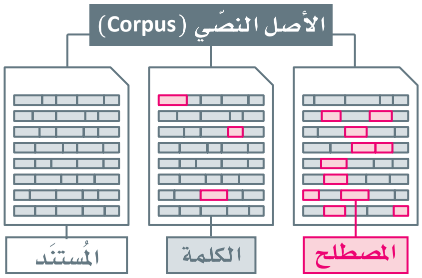
تكرار الكلمة في المُستنَد لا يُعدُّ دومًا تمثيلًا دقيقًا لأهميتها. الطريقة المُثلى لتمثيل التكرار هي المقياس الشهير لتكرار المصطلح - تكرار المُستنَد العكسي (TF-IDF). يستخدِم هذا المقياس صيغة رياضية بسيطة لتحديد أهمية الرموز مثل: الكلمات أو العبارات في المُستنَد بناءً على عاملين:

- تكرار الرمز في المُستنَد، بقياس عدد مرات ظهوره في المُستنَد مقسومًا على إجمالي عدد الرموز في جميع المُستنَدات.
- تكرار المُستنَد العكسي للرمز، المحسوب بقسمة إجمالي عدد المُستنَدات في مجموعة البيانات على عدد المُستنَدات التي تحتوي على الرمز.

العامل الأول يتجنب المبالغة في تقدير أهمية المصطلحات التي تظهر في الوثائق الأطول، أمّا العامل الثاني فيستبعد المصطلحات التي تظهر في كثير من المُستنَدات، مما يساعد على إثبات حقيقة أن بعض الكلمات هي أكثر شيوعًا من غيرها.

**تكرار المصطلح - تكرار المستند العكسي**
**Term Frequency Inverse Document Frequency (TF-IDF):**

تكرار المصطلح - تكرار المُستنَد العكسي هو طريقة تُستخدم لتحديد أهمية الرموز في المُستنَد.




شكل 3.13: الكلمات والمصطلحات الواردة في المستند

$$\text{تكرار المُستنَد العكسي} = \frac{\text{عدد المُستنَدات في الأصل النصّي}}{\text{عدد المُستنَدات التي تحتوي على المصطلح}}$$

$$\text{تكرار المصطلح} = \frac{\text{عدد مرات ظهور المصطلح في المُستنَد}}{\text{عدد الكلمات في المُستنَد}}$$

القيمة = تكرار المصطلح × تكرار المُستنَد العكسي

### أداة TfidfVectorizer

توفر مكتبة **سكليرن** (Sklearn) أداة تدعم هذا النوع من البرمجة الاتجاهية **لتكرار المصطلح-تكرار المُستنَد العكسي** (TF-IDF). يمكن استخدام أداة TfidfVectorizer لتمثيل عبارة باستخدام المتَّجهات.

```
from sklearn.feature_extraction.text import TfidfVectorizer

# Train a TF-IDF model with the IMDb training dataset
vectorizer_tf = TfidfVectorizer(min_df=10)
vectorizer_tf.fit(X_train_text_annotated)
X_train_tf = vectorizer_tf.transform(X_train_text_annotated)
```


وزارة التعليم
Ministry of Education
2025 - 1447

يمكن الآن إدخال أداة التمثيل بالمتَّجَهات في مُصنِّف بايز الساذج لبناء خط أنابيب نموذج تنبُّؤ جديد وتطبيقه على بيانات اختبار IMDb:

```
# train a new Naive Bayes Classifier on the newly vectorized data.
model_tf =MultinomialNB()
model_tf.fit(X_train_v2, Y_train)

# create a new prediction pipeline.
prediction_pipeline_tf = make_pipeline(vectorizer_tf, model_tf)

# get predictions using the new pipeline.
predictions_tf = prediction_pipeline_tf.predict(X_test_text_annotated)

# print the achieved accuracy.
accuracy_score(Y_test, predictions_tf)
```

```
0.8858
```

يحقق خط الأنابيب الجديد دقة تصل إلى 88.58%، وهو تحسُّن كبير بالمقارنة مع الدقة السابقة التي وصلت إلى 84.68%. يمكن الآن استخدام النموذج المُحسَّن لإعادة النظر في مثال الاختبار الذي تم تصنيفه بشكل خاطئ بواسطة النموذج الأول:

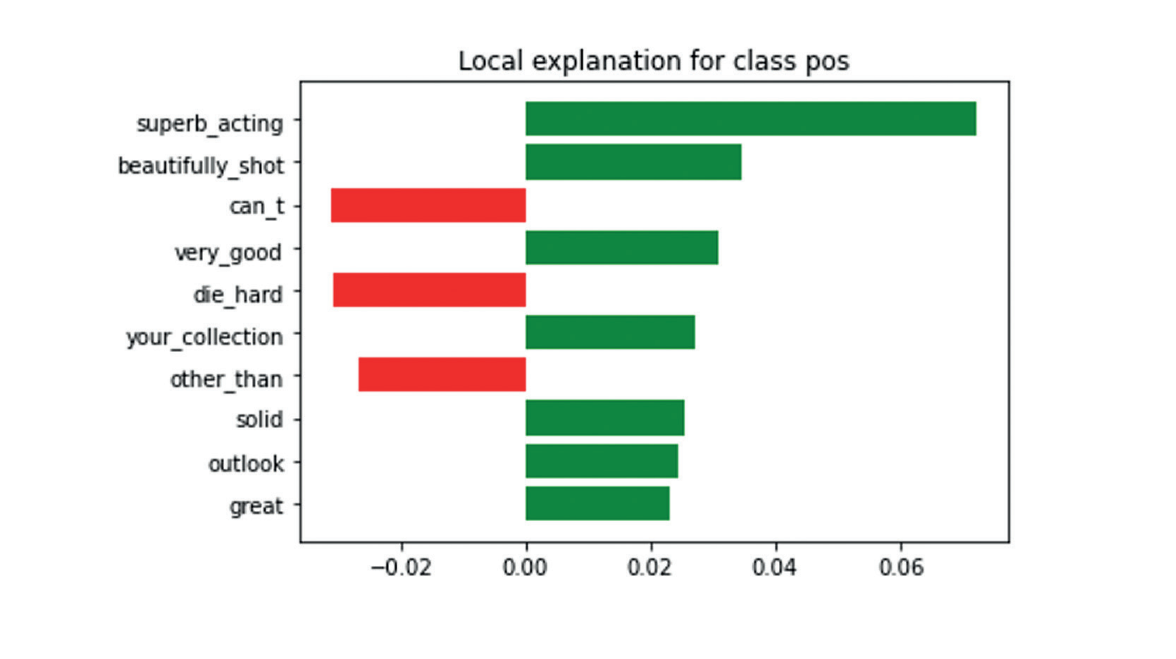
```
# get the review example that confused the previous algorithm
mistake_example_annotated=X_test_text_annotated[4600]

print('\nReview:',mistake_example_annotated)

# get the correct labels of this example.
print('\nCorrect Label:', class_names[Y_test[4600]])

# get the prediction probabilities for this example.
print('\nPrediction Probabilities for neg, pos:',prediction_pipeline_
tf.predict_proba([mistake_example_annotated]))
```

```
Review: i_personally thought the movie was_pretty good very_good acting by tadanobu_
asano of ichi_the_killer fame i really can_t say much about the story but there_were
parts that confused me a little_too much and overall i_thought the movie was just too
lengthy other_than that however the movie contained superb_acting great fighting and
a lot of the locations were beautifully_shot great effects and a lot of sword play
another solid effort by tadanobu_asano in my_opinion well i really can_t say anymore
about the movie but if_you re only outlook on asian_cinema is crouching_tiger hidden_
dragon or house of flying_daggers i_would suggest_you trying to rent_it but if_you re
a die_hard asian_cinema fan i_would say this has to be in your_collection very_good
japanese film

Correct Label: pos

Prediction Probabilities for neg, pos: [[0.32116538 0.67883462]]
```

وزارة التعليم
Ministry of Education
2025 - 1447

يتنبأ خط الأنابيب الجديد بشكل صحيح بالقيمة الإيجابية لهذا التقييم. يَستخدِم المقطع البرمجي التالي مُفسِّر **النموذج المحايد المحلي القابل للتفسير والشرح** (LIME) لتفسير المنطق وراء هذا التنبؤ:

```
# create an explainer.
explainer_tf = LimeTextExplainer(class_names=class_names)

# explain the prediction  of the second pipeline for this example.
exp = explainer_tf.explain_instance(mistake_example_annotated, prediction_
pipeline_tf.predict_proba, num_features=10)

# visualize the results.
fig = exp.as_pyplot_figure()
```




شكل 3.14: تأثير الكلمة في مزيج تكرار المصطلح- تكرار المُستنَد العكسي ومصنّف بايز الساذج

تؤكد النتائج أن خط الأنابيب الجديد يتبع منطقًا أكثر ذكاءً. فهو يُحدد بشكل صحيح المشاعر الإيجابية للعبارات مثل: beautifully_shot (لقطة _ جميلة)، و superb_acting (تمثيل_رائع)، وvery good (جيد جدًا)، ولا يمكن تضليله باستخدام الكلمات التي جعلت خط الأنابيب الأول يتنبأ بنتائج خاطئة.

يمكن تحسين أداء خط الأنابيب لنموذج التنبُّؤ بطرائق متعددة، بإستبدال مصنف بايز البسيط بطرائق أكثر تطورًا مع ضبط متغيراتها لزيادة احتمالاتها. وثمَّة خيار آخر يتلخص في استخدام تقنيات البرمجة الاتجاهية البديلة التي لا تستند إلى تكرار الرمز، مثل تضمين الكلمات والنصوص، وسيُستعرض ذلك في الدرس التالي.


وزارة التعليم
Ministry of Education
2025 - 1447

# تمرينات

**1**

| حدِّد الجملة الصحيحة والجملة الخاطئة فيما يلي: | صحيحة | خاطئة |
|---|---|---|
| 1. في التعلُّم الموجَّه تُستخدم مجموعات البيانات المُعنوَنة لتدريب النموذج. | ○ | ○ |
| 2. البرمجة الاتجاهية هي تقنية لتحويل البيانات من تنسيق متَّجَه رقمي إلى بيانات أولية. | ○ | ○ |
| 3. تتطلب المصفوفة المتباعدة ذاكرة أقل بكثير من المصفوفة الكثيفة. | ○ | ○ |
| 4. تُستخدم خوارزمية مُصنِّف بايز الساذج لبناء خط أنابيب التنبؤ. | ○ | ○ |
| 5. تكرار الكلمة في المُستنَد يُعدُّ التمثيل الدقيق الوحيد لأهمية هذه الكلمة. | ○ | ○ |

**2** اشرح لماذا تتطلب المصفوفة الكثيفة مساحة من الذاكرة أكبر من المصفوفة المتباعدة.

**3** حلِّل كيف يُستخدَم العاملان الرِّياضيّان في تكرار المصطلح- تكرار المُستنَد العكسي (TF-IDF) لتحديد أهمية الكلمة في النص.



وزارة التعليم
Ministry of Education
2025 - 1447

4 لديك X_train_text وهي عبارة عن مصفوفة numPy تتضمن مستندًا واحدًا في كل صف. لديك كذلك مصفوفة ثانية Y_train تتضمن قيم المُستنَدات في X_train_text. أكمل المقطع البرمجي التالي بحيث يمكن استخدام تكرار المصطلح- تكرار المُستنَد العكسي (TF-IDF) لتمثيل البيانات بالمتَّجَهات، وتدريب نموذج تصنيف MultinomialNB على الإصدار المُمَثَّل بالمتَّجَهات، ثم تجميع أداة التمثيل بالمتَّجَهات ونموذج التصنيف في خط أنابيب تنبؤ واحد:

```
from ______________.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline
from sklearn.feature_extraction.text import _______________
vectorizer = _____________(min_df=10)
vectorizer.fit(_________________) # fits the vectorizer on the training data
X_train = vectorizer._____________(X_train_text) # uses the fitted vectorizer to vectorize the data
model_MNB=MultinomialNB() # a Naive Bayes Classifier
model_MNB.fit(X_train, _________________) # fits the classifier on the vectorized training data
prediction_pipeline = make_pipeline(_________________, _________________)
```

5 أكمل المقطع البرمجي التالي بحيث يمكنه بناء مُفسِّر نصوص النموذج المحايد المحلي القابل للتفسير والشرح (LIME) لخط أنابيب التنبؤ الذي قمت ببنائه في التمرين السابق، واستخدِم المُفسِّر لتفسير التنبؤ على مثال لنصٍ آخر.

```
from ______________ import LimeTextExplainer
text_example="I really enjoyed this movie, the actors were excellent"
class_names=['neg','pos'] # creates a local explainer for explaining individual predictions
explainer = ______________(class_names=class_names) # explains the prediction for this example
exp = explainer.__________(text_example.lower(),prediction_pipeline.___________,
                _________________=10) # focuses the explainer on the 10 most influential features
print(exp.________________) # prints the words with the highest influence on the prediction
```


وزارة التعليم
Ministry of Education
2025 - 1447

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الثاني

# التعلُّم غير الموجَّه

## استخدام التعلُّم غير الموجَّه لفهم النصوص

## Using Unsupervised Learning to Understand Text

التعلُّم غير الموجَّه هو نوع من تعلُّم الآلة، يستخدِم فيه النموذج بيانات غير مُعَنونة، حيثُ يُقدِّم له مجموعة من الأمثلة التي يتولى البحث فيها عن الأنماط والعلاقات بين البيانات من تلقاء نفسه. وفي سياق فهم النص، يمكن استخدام التعلُّم غير الموجَّه في تحديد الهياكل والأنماط الكامنة ضمن مجموعة بيانات المُستنَدات النصية. هناك العديد من التقنيات المختلفة التي يمكن استخدامها في التعلُّم غير الموجَّه للبيانات النصية، بما في ذلك **خوارزميات التجميع** (Clustering Algorithms)، و**تقنيات تقليص الأبعاد** (Dimensionality Reduction Techniques)، و**النماذج التوليدية** (Generative Models). تُستخدم خوارزميات التجميع لضم المُستنَدات المتشابهة معًا، بينما تُستخدم تقنيات تقليص الأبعاد لتقليص أبعاد البيانات وتحديد الخصائص الهامة. ومن ناحية أخرى، تُستخدم النماذج التوليدية لتعلُّم التوزيع الأساسي للبيانات وتوليد نص جديد مشابه لمجموعة البيانات الأصلية.

### خوارزميات التجميع Clustering Algorithms

يمكن لخوارزميات التجميع تجميع العملاء المتشابهين استنادًا إلى السلوكيات أو الديموغرافيا، أو سجل المشتريات؛ لأغراض التسويق المُستهدَف وزيادة معدلات الاحتفاظ بالعملاء.

### تقنيات تقليص الأبعاد

### Dimensionality Reduction Techniques

تُستخدم تقنيات تقليص الأبعاد في ضغط الصورة لتقليل عدد وحدات البيكسل فيها، مما يساعد على تقليص حجم البيانات اللازمة لتمثيلها مع الحفاظ على خصائصها الرئيسة.

### النماذج التوليدية Generative Models

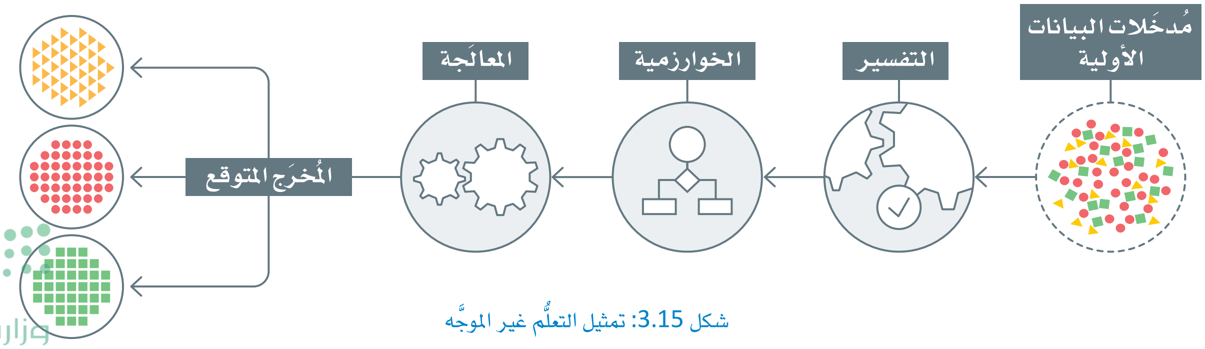
تُستخدم النماذج التوليدية في تطبيقات الكشف عن الاختلاف؛ حيث تُحدِّد الاختلافات في البيانات بتعلُّم الأنماط الطبيعية للبيانات باستخدام النموذج التوليدي.

**التعلُّم غير الموجَّه (Unsupervised Learning):**

في التعلُّم غير الموجَّه، يُزوَّد النموذج بكميات كبيرة من البيانات غير المُعنوَنة ويتوجب عليه البحث عن الأنماط في البيانات غير المُتراكبة من خلال الملاحظة والتجميع.

**تقليص الأبعاد (Dimensionality Reduction):**

تقنية تقليص الأبعاد هي إحدى تقنيات تعلُّم الآلة وتحليل البيانات المُستخدَمة لتقليص عدد الخصائص (الأبعاد) في مجموعة البيانات مع الاحتفاظ بأكبر قدر ممكن من المعلومات.




شكل 3.15: تمثيل التعلُّم غير الموجَّه

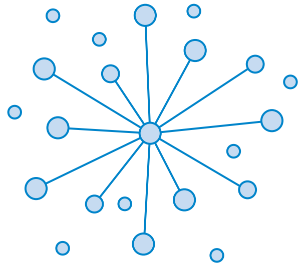


وزارة التعليم
Ministry of Education
2025 - 1447

وإحدى المزايا الرئيسة لاستخدام التعلُّم غير الموجَّه هي أنه يمكن استخدامه للكشف عن الأنماط والعلاقات التي قد لا تبدو واضحة على الفور للمراقب البشري. وقد يكون هذا مفيدًا بشكل خاص في فهم مجموعات البيانات الكبيرة المكونة من النصوص غير المُتراكبة، حيث يكون التحليل اليدوي غير عملي. في هذه الوحدة، ستَستخدم مجموعة بيانات متوافرة للعامّة من المقالات الإخبارية من **هيئة الإذاعة البريطانية** (BBC) بواسطة **جرين وكوننجهام** (Greene & Cunningham، 2006) لتوضيح بعض التقنيات الرئيسة للتعلُّم غير الموجَّه. يُستخدم المقطع البرمجي التالي لتحميل مجموعة البيانات، المُنظّمة في خمسة مجلدات إخبارية مختلفة تمثل مقالات من أقسام إخبارية مختلفة، هي: الأعمال التجارية، والسياسة، والرياضة، والتقنية، والترفيه. لن تستخدِم القيم الخمسة في توجيه أيٍ من الخوارزميات المُستخدَمة في هذه الوحدة. وبدلًا من ذلك، ستُستخدم فقط لأغراض التصوير والمصادقة. يتضمن كل مجلد إخباري مئات الملفات النصية، وكل ملف يتضمن محتوى مقالة واحدة محددة. وقد حُمّلت مجموعة البيانات بالفعل إلى **مفكرة جوبيتر** (Jupyter Notebook) وستقوم لبنة التعليمات البرمجية بفتح واستخراج كل المُستندَات والقيم المطلوبة في تركيبتين لبيانات القوائم، على التوالي.

**العنقود (Cluster):**

العنقود هو مجموعة من الأشياء المتشابهة. وفي تعلُّم الآلة، يشير التجميع (Clustering) إلى عملية تجميع البيانات غير المُعنوَنة في عناقيد متجانسة.



شكل 3.16: تمثيل عنقود

BBC open dataset
https://www.kaggle.com/datasets/shivamkushwaha/bbc-full-text-document-classification
D. Greene and P. Cunningham. "Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering",Proc. ICML 2006. All rights, including copyright, in the content of the original articles are owned by the BBC.

```
# used to list all the files and subfolders in a given folder
from os import listdir
# used for generating random number
import random shuffling lists

bbc_docs=[] # holds the text of the articles
bbc_labels=[] # holds the news section for each article

for folder in listdir('bbc'): # for each news-section folder
    for file in listdir('bbc/'+folder): # for each text file in this folder

        # open the text file, use encoding='utf8' because articles may include non-ascii characters
        with open('bbc/'+folder+'/'+file,encoding='utf8',errors='ignore') as f:
            bbc_docs.append(f.read()) # read the text of the article and append to the docs list
        # use the name of the folder (news section) as a label for this doc
        bbc_labels.append(folder)
# shuffle the docs and labels lists in parallel
merged = list(zip(bbc_docs, bbc_labels)) # link the two lists
random.shuffle(merged) # shuffle them in parallel (with the same random order)
bbc_docs, bbc_labels = zip(*merged) # separate them again into individual lists.
```

# تجميع المُستنَدات Document Clustering

الآن بعد تحميل مجموعة البيانات فإن الخطوة التالية هي تجربة عدة طرائق غير موجَّهة، ومنها: التجميع الذي يُعدّ الطريقة غير الموجَّهة الأكثر شهرة في هذا النطاق. وبالنّظر إلى مجموعة من المُستنَدات غير المُعنوَنة، سيكون الهدف هو تجميع الوثائق المتشابهة معًا، وفي الوقت نفسه الفصل بين الوثائق غير المتشابهة.

**تجميع المُستنَدات (Document Clustering):**

تجميع المُستنَدات هو طريقة تُستخدم لتجميع المُستنَدات النصيّة في عناقيد بناءً على تشابه محتواها.

**جدول 3.2: العوامل التي تُحدد جودة النتائج**

| | |
|---|---|
| 1 | **طريقة تمثيل البيانات بالمتَّجَهات:** على الرغم من أن تقنية **تكرار المصطلح- تكرار المُستنَد العكسي** (TF-IDF) أثبتت كفاءتها وفعاليتها في هذا المجال، إلا أنّك ستتعرف في هذه الوحدة على مزيد من البدائل الأكثر تطورًا وتعقيدًا. |
| 2 | **التعريف الدقيق للتشابه بين مستند وآخر:** بالنسبة للبيانات النصيّة المُمثلة بالمتَّجَهات، تكون مقاييس المسافة الإقليدية وجيب التمام هما الأكثر شيوعًا، وسيُستخدم الأول في الأمثلة المشروحة في هذه الوحدة. |
| 3 | **عدد العناقيد المُختارة:** يوفر **التجميع التكتلي** (Agglomerative Clustering - AC) طريقة واضحة لتحديد العدد المناسب من العناقيد ضمن مجموعة محددة من البيانات، وهو التحدي الرئيس الذي يواجه مهام التجميع. |

## تحديد عدد العناقيد

## Selecting the Number of Clusters

تحديد العدد الصحيح للعناقيد هو خطوة ضرورية ضمن مهام التجميع. للأسف، تعتمد الغالبية العظمى من خوارزميات التجميع على المُستخدِم في تحديد عدد العناقيد الصحيحة ضمن المُدخَلات. ربما يكون للعدد المحدد تأثيرًا كبيرًا على جودة النتائج وقابليتها للتفسير، ولكن هناك العديد من المقاييس أو المؤشرات التي يمكن استخدامها لتحديد عدد العناقيد.

- إحدى الطرائق الشائعة هي استخدام مقياس **التراص** (Compactness). يمكن القيام بذلك عن طريق حساب مجموع المسافات بين النقاط ضمن كل عنقود، وتحديد عدد العناقيد الذي يقلل من هذا المجموع إلى الحد الأدنى.
- هناك طريقة أخرى تتلخص في مقياس **الفصل** (Separation) بين العناقيد، مثل متوسط المسافة بين النقاط في العناقيد المختلفة، وبناء عليه، يتم تحديد عدد العناقيد الذي يرفع من هذا المتوسط.

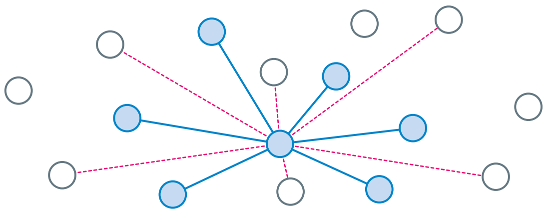
وبشكل عملي، غالبًا ما تتعارض المنهجيات المذكورة بالأعلى مع بعضها من حيث التوصية بأرقام مختلفة، ويمثّل هذا تحدّيًا مشتركًا عند التعامل مع البيانات النصية بشكلٍ خاص، فعادةً ما يصعُب تمييز تركيبها.

**المسافة الإقليدية (Euclidean Distance):**

المسافة الإقليدية هي مسافة الخط المستقيم بين نقطتين في فضاء متعدد الأبعاد. وتُحسب بالجذر التربيعي لمجموع مربعات الفروقات بين الأبعاد المناظرة للنقاط. تُستخدم المسافة الإقليدية في التجميع لقياس التشابه بين نقطتي بيانات.

**مسافة جيب التمام (Cosine Distance):**

تُستخدم مسافة جيب التمام لقياس التشابه في جيب التمام بين نقطتي البيانات. فهي تحسب جيب تمام الزاوية بين متَّجَهين يمثلان نقاط البيانات، وتُستخدَم عادةً في تجميع البيانات النصيّة. وتقع قيمة جيب التمام بين 1- و 1؛ حيث تشير القيمة 1- إلى الاتجاه العكسي، بينما تشير القيمة 1 إلى الاتجاه نفسه.



شكل 3.17: آلة حساب المسافات بين النقاط



وزارة التعليم
Ministry of Education
2025 - 1447

في التعلُّم غير الموجَّه، يشير عدد العناقيد إلى عدد المجموعات أو التصنيفات التي تنقسم إليها البيانات بواسطة الخوارزمية. ويُعدُّ تحديد عدد العناقيد الصحيح أمرًا مهمًا؛ لأنه يؤثر على دقة النتائج وقابليتها للتفسير. إذا كان عدد العناقيد كبيرًا للغاية، فإنّ المجموعات ستكون محدّدةً جدًا ودون معنى. في حين أنه إذا كان عدد العناقيد منخفضًا للغاية، فإنّ المجموعات ستكون ممتدة على نطاق واسع جدًا، ولن تستنبط التركيب الأساسي للبيانات. من الضروري تحقيق التوازن بين توفير عددٍ كافٍ من العناقيد لاستنباط أنماط ذات معنى، وألا تكون كثيرة في الوقت نفسه بالقدر الذي يجعل النتائج مُعقدة للغاية وغير مفهومة.

**التجميع الهرمي (Hierarchical Clustering):**

التجميع الهرمي هو خوارزمية التجميع المُستخدَمة لتجميع البيانات في عناقيد بناءً على التشابه. في التجميع الهرمي، تُنظّم نقاط البيانات في تركيب يشبه الشجرة، حيث تكون كل عُقدة بمثابة عنقود، وتكون العُقدة الأم هي نقطة التقاء العُقد المتفرعة منها.

يُستخدَم المقطع البرمجي التالي لاستيراد مكتبات محددة تُستخدَم في التجميع الهرمي من بدايته حتى نهايته:

```
# used for tfi-df vectorization, as seen in the previous unit
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import AgglomerativeClustering # used for agglomerative clustering

# used to visualize and support hierarchical clustering tasks
import scipy.cluster.hierarchy as hierarchy

# set the color palette to be used by the 'hierarchy' tool.
hierarchy.set_link_color_palette
(['blue','green','red','yellow','brown','purple','orange','pink','black'])

import matplotlib.pyplot as plt # used for general visualizations
```

## البرمجة الاتجاهية للنصوص Text Vectorization

تتطلب العديد من طرائق التعلُّم غير الموجَّه تمثيل النصّ الأوليّ بالمتَّجَهات في تنسيق رقميّ، كما تمّ عرضه في الوحدة السابقة، ويستخدِم المقطع البرمجي التالي أداة TfidfVectorizer التي اُستخدمت في الدرس السابق لهذا الغرض:

```
vectorizer = TfidfVectorizer(min_df=10) # apply tf-idf vectorization, ignore words that
appear in more than 10 docs.

text_tfidf=vectorizer.fit_transform(bbc_docs) # fit and transform in one line

text_tfidf
```

```
<2225x5867 sparse matrix of type '<class 'numpy.float64'>'
	with 392379 stored elements in Compressed Sparse Row format>
```

الآن تحوَّلت بيانات النص إلى تنسيق رقمي متباعد كما اُستخدمت في الدرس السابق.

يَستخدم المقطع البرمجي التالي أداة TSENVisualizer من مكتبة yellowbrick لإسقاط وتصوير النصوص المُمثلة بالمُتَّجَهات في فضاء ثنائي الأبعاد:

```
%%capture
!pip install yellowbrick
from yellowbrick.text import TSNEVisualizer
```

## تقليص الأبعاد Dimensionality Reduction

يكون تقليص الأبعاد مفيدًا في العديد من التطبيقات مثل:

- **تصوير البيانات عالية الأبعاد:** من الصعب تصوير البيانات في فضاء عالي الأبعاد، ولذلك تُقلّص الأبعاد ليسهل تصوير البيانات وفهمها في هذه الحالة.
- **تبسيط النموذج:** النموذج ذو الأبعاد الأقل يكون أبسط وأسهل فهمًا، ويستغرق وقتًا أقل في عملية التدريب.
- **تحسين أداء النموذج:** يُساعد تقليص الأبعاد في التخلص من التشويش وتكرار البيانات، مما يُحسّن أداء النموذج.

**تضمين المجاور العشوائي الموزع على شكل T t-Distributed Stochastic Neighbor Embedding (T-SNE):**

خوارزمية تضمين المجاور العشوائي الموزَّع على شكل T (T-SNE) هي خوارزمية تعلُّم الآلة غير الموجَّه المُستخدَمة لتقليص الأبعاد.

### جدول 3.3: تقنيات تقليص الأبعاد

| التقنية | الوصف | مثال التطبيق العملي |
|---|---|---|
| تحديد الخصائص (Feature Selection) | تحديد الخصائص يتضمن تحديد مجموعة فرعية من الخصائص الرئيسة. | تحتوي مجموعات البيانات الطبية على مئات من أعمدة البيانات ذات الصلة بحالة المريض. يمكن لعدد قليل من هذه الخصائص مساعدة النموذج في التشخيص السليم لحالة المريض، بينما تكون السمات الأخرى غير مرتبطة بالتشخيص وقد تُشتت النموذج، وتحديد الخصائص يتجاهل كل الخصائص بإستثناء الأكثر تميزًا منها. |
| تحويل الخصائص (Feature Transformation) | يتضمن تحويل الخصائص تجميع الخصائص الأصلية أو تحويلها لإنشاء مجموعة جديدة من الخصائص، واستبدال الخصائص الرئيسة إذا لم تكن هناك حاجة إليها. | إذا توقَّع النموذج إقامة المريض في المستشفى، يمكن إنشاء خصائص إضافية للنموذج باستخدام الخصائص الحالية لسجلات الحالة الطبية للمريض. على سبيل المثال، حساب عدد الفحوصات المخبرية المطلوبة على مدار الأسبوع الماضي، أو عدد الزيارات على مدار الشهر الماضي. وهناك مثال آخر، وهو: حساب مساحة المستطيل بإستخدام ارتفاعه وعرضه. |
| التعلُّم المتشعِّب (Manifold Learning) | تقنيات التعلُّم المتشعِّب، مثل **تضمين المجاور العشوائي الموزَّع على شكل T** (T-SNE) و**التقريب والإسقاط المنتظم المتشعِّب** (Uniform Manifold Approximation and Projection –UMAP) هي تقنيات التعلُّم غير الموجَّه التي تهدف إلى الحفاظ على تركيب البيانات في الفضاء منخفض الأبعاد. | يمكن لهذه التقنيات تحويل صورة عالية الأبعاد إلى فضاء منخفض الأبعاد مع الحفاظ على الخصائص والتركيب الأساسيين لها. ونظرًا لأن هذا يقلص من المساحة المطلوبة، فإنه يمكن تخزين وإرسال هذا التمثيل وإعادة بناء الصورة الأصلية مع خسارة أقل قدر من المعلومات. |

وزارة التعليم
Ministry of Education
2025 - 1447

إحدى الخصائص الرئيسة لتقنية **تضمين المجاور العشوائي الموزَّع على شكل T** (T-SNE) هي محاولة الحفاظ على التركيب المحلي للبيانات قدر الإمكان، حتى تتقارب نقاط البيانات الشبيهة في التمثيل منخفض الأبعاد، ويتحقق ذلك بتقليص التباعد بين التوزيعين المحتملين: توزيع البيانات عالية الأبعاد، وتوزيع البيانات منخفضة الأبعاد.

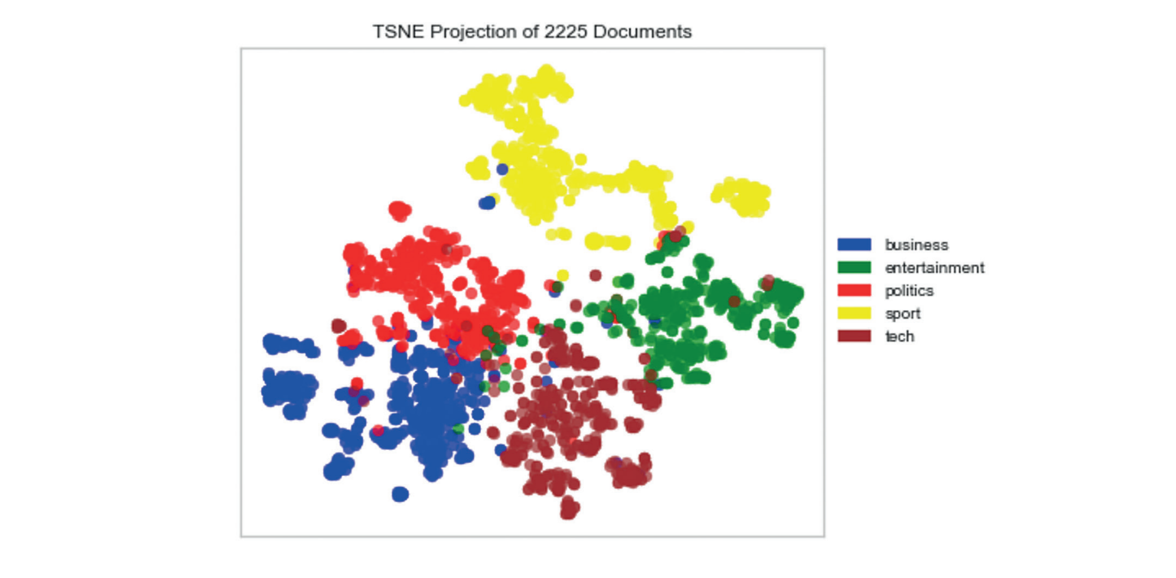
مجموعة بيانات هيئة الإذاعة البريطانية المُمثلة بالمتَّجَهات تُصنَّف بالتأكيد كبيانات عالية الأبعاد، لأنها تتضمن بُعدًا مستقلًا أي **عمودًا** (Column) لكل كلمة فريدة تظهر في البيانات. يُحسب العدد الإجمالي للأبعاد كما يلي:

```
print('Number of unique words in the BBC documents vectors:',
                               len(vectorizer.get_feature_names_out()))
```

```
Number of unique words in the BBC documents vectors: 5867
```

يُستخدَم المقطع البرمجي التالي لإسقاط 5,867 بُعدًا في محورين فقط وهما محوري X و Y في الرسم البياني.
يُستخدَم المقطع البرمجي التالي لتصميم مُخطَّط الانتشار حيث يمثل كل لون أحد الأقسام الإخبارية الخمسة.

```
tsne = TSNEVisualizer(colors=['blue','green','red','yellow','brown'])
tsne.fit(text_tfidf,bbc_labels)
tsne.show();
```



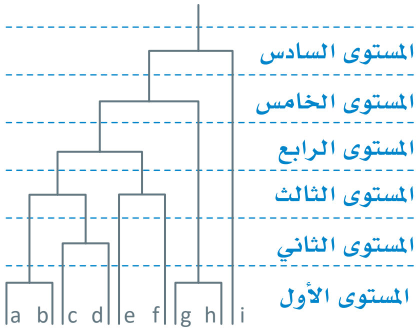

شكل 3.18: إسقاط تضمين المجاور العشوائي الموزَّع على شكل T (T-SNE)

يَستخدِم هذا التصور قيمة ground-truth (بيانات الحقيقة المعتمدة) من **القسم الإخباري** (News Section) في كل مستند للكشف عن انتشار كل قيمة في إسقاط فضاء البرمجة الاتجاهية ثنائي الأبعاد. يوضِّح الشكل أنه على الرغم من ظهور بعض الشوائب في فراغات مُحدَّدة من فضاء البيانات، إلا أن الأقسام الإخبارية الخمسة منفصلة بشكل جيد. وسنستعرض لاحقًا البرمجة الاتجاهية المُحسَّنة للحد من هذه الشوائب.


وزارة التعليم
Ministry of Education
2025 - 1447

## التجميع التكتلي (AC) Agglomerative Clustering

**التجميع التكتلي** (AC) هو الطريقة الأكثر انتشارًا وفعاليةً في هذا الفضاء، فمن خلالها يمكن التغلّب على هذا التحدي بتوفير طريقة واضحة لتحديد العدد المناسب من العناقيد. يستند **التجميع التكتلي** (AC) إلى منهجية التصميم من أسفل إلى أعلى، حيث تبدأ بحساب المسافة بين كل أزواج نقاط البيانات، ثم اختيار النقطتين الأقرب ودمجهما في عنقود واحد. تتكرر هذه العملية حتى تُدمج كل نقاط البيانات في عنقود واحد، أو حتى الوصول إلى العدد المطلوب من العناقيد.



شكل 3.19: التجميع التكتلي (AC)

## *fx* دالة Linkage()

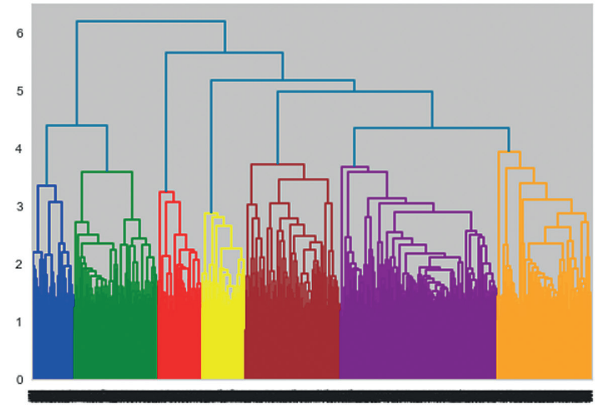
تُنفِذ لغة البايثون **التجميع التكتلي** (AC) باستخدام دالة linkage().

يجب توفير متغيرين لدالة linkage():

- البيانات النصيّة المُمثلة بالمتَّجَهات، ويمكن استخدام دالة toarray() لتحويل البيانات إلى تنسيق كثيف يمكن لهذه الدالة أن تتعامل معه.
- مقياس المسافة الذي يجب استخدامه لتحديد العناقيد التي ستُدمج أثناء عملية التجميع التكتلي. تتوفر عدة خيارات من مقاييس المسافة للاختيار من بينها وفقًا لمتطلبات وتفضيلات المُستخدِم، مثل المسافة **الإقليدية** (Euclidian)، ومسافة **مانهاتن** (Manhattan)... إلخ، ولكن في هذا المشروع ستستخدِم طريقة **وارد** (**ward**) القياسية.

يستخدِم المقطع البرمجي التالي دالة linkage() من الأداة **الهرمية** (Hierarchy) الواردة بالأعلى لتطبيق هذه العملية على بيانات هيئة الإذاعة البريطانية المُمثَّلة بالمتَّجَهات:

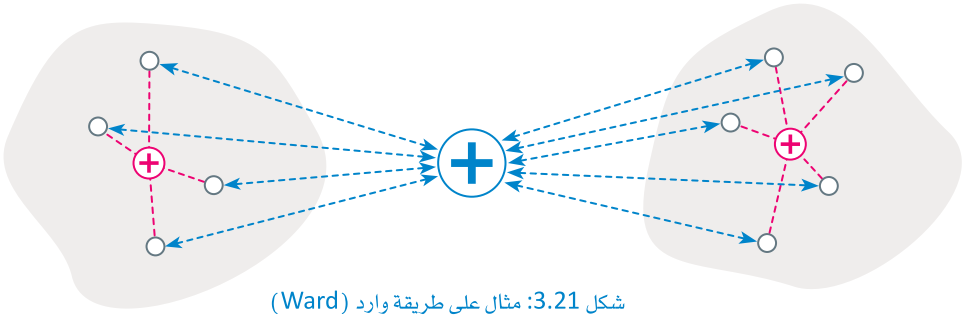
```
plt.figure() # create a new empty figure

# iteratively merge points and clusters until all points belong to a single cluster
# return the linkage of the produced tree
linkage_tfidf=hierarchy.linkage(text_tfidf.toarray(),method='ward')

# visualize the linkage
hierarchy.dendrogram(linkage_tfidf)

# show the figure
plt.show()
```



شكل 3.20: الرسم الشجري الهرمي لبيانات هيئة الإذاعة البريطانية

## مسافة وارد Ward Distance

يَستخدِم المثال أعلاه طريقة **وارد** (Ward) القياسية لقياس المسافة للمتغير الثاني. تستند **مسافة وارد** (Ward) إلى مفهوم التباين داخل العنقود، وهو مجموع المسافات بين النقاط في العنقود. في كل تكرار، تُقيِّم الطريقة كل عملية دمج ممكنة بحساب التباين داخل العنقود قبل عمليّة الدمج وبعدها، ثم تبدأ عملية الدمج التي تحقِّق أقل ارتفاع في التباين. أظهرت **مسافة وارد** (Ward) نتائج جيدة في معالجة البيانات النصيّة، بالرغم من وجود العديد من الخيارات الأخرى.



شكل 3.21: مثال على طريقة وارد (Ward)

**الرسم الشجري (Dendrogram):**

الرسم الشجري هو رسم تخطيطي تفرعي يوضِّح العلاقة الهرمية بين البيانات، ويأتي عادة في صورة أحد مُخرَجات التجميع الهرمي.

الرسم الشجري في الشكل 3.20 يعرض طريقة واضحة لتحديد عدد العناقيد. في هذا المثال، تقترح المكتبة استخدام 7 عناقيد، مع تمييز كل عنقود بلون مختلف. قد يتبنى المُستخدِم هذا المُقترح أو يَستخدِم الرسم الشجري لاختيار رقم مختلف. على سبيل المثال، دُمِّج اللونين الأزرق والأخضر في آخر خطوة مع مجموعة العناقيد لكل الألوان الأخرى. وهكذا، سيؤدي اختيار 6 عناقيد إلى دمج اللونين الأرجواني والبرتقالي، بينما اختيار 5 عناقيد سيؤدي إلى دمج اللونين الأزرق والأخضر.

يتبنى المقطع البرمجي التالي مقترحات الأداة ويَستخدِم أداة التجميع التكتلي من مكتبة **سكليرن** (Sklearn) لتقسيم المُخطَّط الشجري بعد إنشاء العناقيد السبع:

```
AC_tfidf=AgglomerativeClustering(linkage='ward',n_clusters=7) # prepare the tool, set the number of clusters.

AC_tfidf.fit(text_tfidf.toarray()) # apply the tool to the vectorized BBC data.

pred_tfidf=AC_tfidf.labels_ # get the cluster labels.

pred_tfidf
```

```
array([6, 2, 4, ..., 6, 3, 5], dtype=int64)
```

لاحظ أن قيمة ground-truth (بيانات الحقيقة المعتمدة) من **القسم الإخباري** (News Section) في كل مستند لم تُستخدَم على الإطلاق في هذه العملية. وبدلًا من ذلك، عولجت عملية التجميع استنادًا إلى نص محتوى كل وثيقة على حده. إنَّ قيم بيانات الحقيقة المعتمدة مفيدة في التطبيق العملي، فهي تتيح التحقق من صحة نتائج التجميع. وقيم بيانات الحقيقة المعتمدة الحالية موجودة في قائمة bbc_labels (قيم_ هيئة الإذاعة البريطانية).

يَستخدِم المقطع البرمجي التالي قيم بيانات الحقيقة المعتمدة وثلاثة دوال مختلفة لتسجيل النقاط من مكتبة **سكليرن** (Sklearn) لتقييم جودة تجميع البيانات:

- تكون قيم **مؤشر التجانس** (Homogeneity Score) بين 0 و 1 ويمكن زيادة هذه القيم عندما تكون كل النقاط في كل عنقود لها قيمة بيانات الحقيقة المعتمدة. وبالمثل، يحتوي كل عنقود على نقاط البيانات وحيدة التصنيف.
- تكون قيمة **مؤشر راند المُعدل** (Adjusted Rand Score) بين 0.5- و 1.0 ويمكن زيادة هذه القيم عندما تقع كل نقاط البيانات ذات القيم نفسها في العنقود نفسه وكل نقاط البيانات ذات القيم المختلفة في عناقيد مختلفة.
- تكون قيمة **مؤشر الاكتمال** (Completeness Score) بين 0 و 1 ويمكن زيادة هذه القيمة بتعيين كل نقاط البيانات من تصنيف مُحدد في العنقود نفسه.

```
from sklearn.metrics import homogeneity_score,adjusted_rand_
score,completeness_score

print('\nHomogeneity score:',homogeneity_score(bbc_labels,pred_tfidf))
print('\nAdjusted Rand score:',adjusted_rand_score(bbc_labels,pred_tfidf))
print('\nCompleteness score:',completeness_score(bbc_labels,pred_tfidf))
```

```
Homogeneity score: 0.6224333236569846

Adjusted Rand score: 0.4630492696176891

Completeness score: 0.5430590192420555
```

المؤشر أقرب إلى 1 وهذا يعني أن مجموعة النصوص في العنقود تنتمي إلى قيمة واحدة.

المؤشر أقرب إلى 1 وهذا يعني إنشاء روابط أفضل بين العناقيد والقيم؛ كلٌّ على حدة.

لاستكمال تحليل البيانات، يُعاد تجميع البيانات باستخدام 5 عناقيد، بالتساوي مع العدد الحقيقي لقيم ground-truth (بيانات الحقيقة المعتمدة):

```
AC_tfidf=AgglomerativeClustering(linkage='ward',n_clusters=5)
AC_tfidf.fit(text_tfidf.toarray())
pred_tfidf=AC_tfidf.labels_

print('\nHomogeneity score:',homogeneity_score(bbc_labels,pred_tfidf))
print('\nAdjusted Rand score:',adjusted_rand_score(bbc_labels,pred_tfidf))
print('\nCompleteness score:',completeness_score(bbc_labels,pred_tfidf))
```

```
Homogeneity score: 0.528836079209762

Adjusted Rand score: 0.45628412883628383

Completeness score: 0.6075627851312266
```

نظرًا لقدرة التجميع الهرمي على إيجاد العدد الحقيقي من القيم، وتوفير مؤشر اكتمال أكثر دقة، ستحصل على عملية تجميع أفضل من حيث تمثيل البيانات.

على الرغم من أن نتائج المؤشر تُظهر أن التجميع التكتلي باستخدام البرمجة الاتجاهية **لتكرار المصطلح-تكرار المُستنَد العكسي** (TF-IDF) تحقق نتائج معقولة، إلا أنّه لا يزال بالإمكان تحسين دقة عملية التجميع. سيوضّح القسم التالي كيف يمكن أن نحقق نتائج مبهرة باستخدام تقنيات البرمجة الاتجاهية المُستنِدة على الشبكات العصبية.

# البرمجة الاتجاهية للكلمات باستخدام الشبكات العصبية
## Word Vectorization with Neural Networks

البرمجة الاتجاهية **لتكرار المصطلح-تكرار المُستنَد العكسي** (TF-IDF) تستند إلى حساب تكرار الكلمات ومعالجتها عبر المُستنَدات في مجموعة البيانات. بالرغم من أن هذا يحقق نتائج جيدة، إلا أنّ القيود الكبيرة تعيب الطرائق المستنِدة إلى التكرار. فهي تتجاهل تمامًا العلاقة الدلالية بين الكلمات. على سبيل المثال، على الرغم من أن كلمتي trip (نزهة) وjourney (رحلة) مترادفتان، إلا أنّ البرمجة الاتّجاهيّة المُستندة إلى التّكرار ستتعامل معهما باعتبارهما كلمتان منفصلتان تمامًا ولهما خصائص مستقلة. وبالمثل، بالرغم من أن كلمتي apple (تفاحة) و fruit (فاكهة) مترابطتان دلاليًا؛ لأن التفاح نوع من الفاكهة إلا أنّ ذلك لن يؤخذ بعين الاعتبار أيضًا.

تؤثر هذه القيود كثيرًا على التطبيقات التي تستخدِم هذا النوع من البرمجة الاتجاهية. فكِّر في الجملتين التاليتين:

- I have a very high fever، so I have to visit a doctor (لديّ حمّى شديدة، ويجب عليّ زيارة الطبيب).
- My body temperature has risen significantly، so I need to see a healthcare professional (ارتفعت درجة حرارة جسمي كثيرًا، ويجب عليّ زيارة أخصائي الرعاية الصحية).

بالرغم من أن الجملتين تصفان الحالة نفسها إلا أنهما لا تتشاركان أي كلمات دلالية. ولذلك، ستفشل خوارزميات التجميع المُستنِدة إلى **تكرار المصطلح-تكرار المُستنَد العكسي** (TF-IDF) أو أي برمجة اتجاهية (تستند إلى التكرار) في رؤية التشابه بين الكلمات، ومن المحتمل ألا تضعها في نفس العنقود.

### نموذج الكلمة إلى المتَّجَه Word2Vec

يمكن معالجة هذه القيود بالطرائق التي تأخذ بعين الاعتبار التشابه الدلالي بين الكلمات. إحدى الطرائق الشهيرة المُتبعة في هذا الصدد هي **نموذج الكلمة إلى المتَّجَه** (Word2Vec) التي تَستخدِم بُنية تَستند إلى الشبكات العصبية. يَستند **نموذج الكلمة إلى المتَّجَه** (Word2Vec) إلى فكرة أن الكلمات المتشابهة دلاليًا تُحاط بكلمات مماثلة في السياق نفسه. ولذلك، نجد الشبكات العصبية تستخدِم التضمين الخفي لكل كلمة للتنبؤ بالسياق، مع ضرورة إنشاء الروابط بين الكلمات والتضمينات الشبيهة. عمليًا، يخضع **نموذج الكلمة إلى المتَّجَه** (Word2Vec) للتدريب المُسبَق على ملايين المُستنَدات لتعلُّم التضمين عالي الدقة للكلمات. يمكن تحميل النماذج المُدرَّبة مسبقًا واستخدامها في التطبيقات المُستندة إلى النصوص. يَستخدِم المقطع البرمجي التالي مكتبة **جينسم** (Gensim) لتحميل نموذج شهير مُدرَّب مسبقًا على مجموعة كبيرة جدًا من **أخبار قوقل** (Google News):

```
import gensim.downloader as api
model_wv = api.load('word2vec-google-news-300')
fox_emb=model_wv['fox']
print(len(fox_emb))
```

```
300
```

هذا النموذج يربط كل كلمة بتضمين مكوّن من 300 بُعد.

**الكلمات المُستبعَدة (Stopwords):**

الكلمات المُستبعَدة هي كلمات شائعة في اللغات تُستبعد عادةً أثناء المعالجة المُسبَقة للنصوص ضمن مهام **معالجة اللغات الطبيعية** (NPL) مثل البرمجة الاتجاهية للكلمات. هذه الكلمات تشمل أدوات التعريف، وحروف العطف، وحروف الجر، والكلمات التي لا تكون مفيدة لتحديد معنى النصّ، أو سياقه.

**التضمين (Embedding):**

التضمين يُعبِّر عن الكلمات أو الرموز في فضاء المتَّجَه المستمر حيث ترتبط الكلمات المتشابهة دلاليًا مع النقاط القريبة.

الأبعاد العشرة الأولى للتضمين العددي لكلمة fox (ثعلب) موضحة بالأسفل:

```
fox_emb[:10]
```

```
array([-0.08203125, -0.01379395, -0.3125    , -0.04125977,  0.05493164,
       -0.12988281, -0.10107422, -0.00164795,  0.15917969,  0.12402344],
      dtype=float32)
```

يَستخدِم النموذج تضمينات الكلمات لتقييم درجة التشابه. فكِّر في المثال التالي حيث تُظهر المقارنة بين كلمة car (السيارة) والكلمات الأخرى درجة التشابة من خلال تناقص قيم التشابة. علمًا بأن قيم التشابه تقع دومًا بين 0 و 1.

```
pairs = [
    ('car', 'minivan'),
    ('car', 'bicycle'),
    ('car', 'airplane'),
    ('car', 'street'),
    ('car', 'apple'),
]
for w1, w2 in pairs:
    print(w1, w2, model_wv.similarity(w1, w2))
```

```
car minivan 0.69070363
car bicycle 0.5364484
car airplane 0.42435578
car street 0.33141237
car apple 0.12830706
```

يُمكن استخدام المقطع البرمجي التالي للعثور على الكلمات الخمسة المشابهة لإحدى الكلمات:

```
print(model_wv.most_similar(positive=['apple'], topn=5))
```

```
[('apples', 0.720359742641449), ('pear', 0.6450697183609009),
('fruit', 0.6410146355628967), ('berry', 0.6302295327186584), ('pears',
0.613396167755127)]
```

يُمكن استخدام التصوير في التحقق من صحة تضمينات هذا النموذج المُدرَّب مُسبقًا، ويُمكن تحقيق ذلك عبر:

- تحديد نماذج الكلمات من مجموعة بيانات هيئة الإذاعة البريطانية.
- استخدام **تضمين المجاور العشوائي الموزَّع على شكل** T (T-SNE) لتخفيض التضمين ذي الـ 300 بُعدٍ لكل كلمة إلى نقطة ثنائية الأبعاد.
- تصوير النقاط في مُخطَّط الانتشار في الفضاء ثنائي الأبعاد.

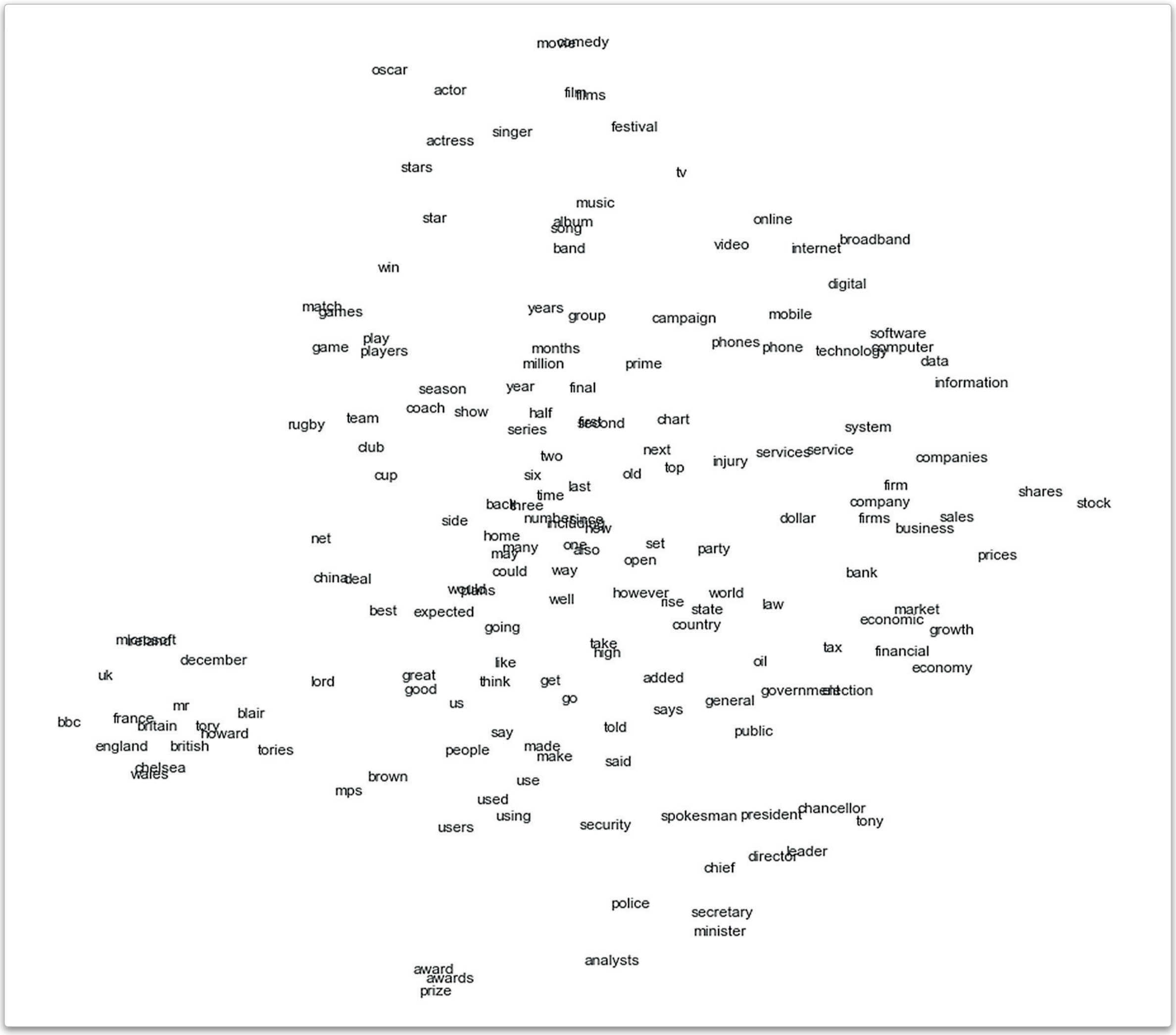
```python
%%capture
import nltk # import the nltk library for nlp.
import re # import the re library for regular expressions.
import numpy as np # used for numeric computations
from collections import Counter # used to count the frequency of elements in a given list
from sklearn.manifold import TSNE # Tool used for Dimensionality Reduction.

# download the 'stopwords' tool from the nltk library. It includes very common words for different languages
nltk.download('stopwords')

from nltk.corpus import stopwords # import the 'stopwords' tool.

stop=set(stopwords.words('english')) # load the set of english stopwords.
```

تُستخدَم الدالة الآتية لاحقًا لتحديد عينة من الكلمات التمثيلية من مجموعة بيانات هيئة الإذاعة البريطانية. يُحدِّد المقطع البرمجي الكلمات الخمسين الأكثر تكرارًا على وجه التحديد من الأقسام الإخبارية الخمسة لهيئة الإذاعة البريطانية مع استثناء **الكلمات المُستبعَدة** (Stopwords) وهي الكلمات الإنجليزية الشائعة جدًا والكلمات التي لم تُضمَّن في **نموذج الكلمة إلى المتَّجَه** (Word2Vec) المُدرَّب مسبقًا.

بعض الكلمات الإنجليزية الشائعة التي تُعدُّ كلمات مُستبعَدة (Stopwords) هي a (أ) و the (ال) و is (يكون) و are (يكونون).

```python
def get_sample(bbc_docs:list,
               bbc_labels:list
               ):

    word_sample=set() # a sample of words from the BBC dataset

    # for each BBC news section
    for label in ['business', 'entertainment', 'politics', 'sport', 'tech']:

        # get all the words in this news section, ignore stopwords.
        # for each BBC doc and for each word in the BBC doc
        # if the word belongs to the label and is not a stopword and is included in the Word2Vec model
        label_words=[word for i in range(len(bbc_docs))
                     for word in re.findall(r'\b\w\w+\b',bbc_docs[i].lower())
                         if bbc_labels[i]==label and
                            word not in stop and
                            word in model_wv]

        cnt=Counter(label_words) # count the frequency of each word in this news section.

        # get the top 50 most frequent words in this section.
        top50=[word for word,freq in cnt.most_common(50)]
        # add the top50 words to the word sample.
        word_sample.update(top50)


    word_sample=list(word_sample) # convert the set to a list.
    return word_sample

word_sample=get_sample(bbc_docs,bbc_labels)
```

وأخيرًا، ستَستخدِم طريقة **تضمين المجاور العشوائي الموزَّع على شكل T** (T-SNE) لتخفيض التضمينات ذات الـ 300 بُعدٍ للكلمات في العينة ضمن النقاط ثنائية الأبعاد. بعدها، تُمثَّل النقاط في مُخطَّط انتشار بسيط.



شكل 3.22: تمثيل الكلمات الأكثر تكرارًا من مجموعة بيانات هيئة الإذاعة البريطانية

يُثبت المُخطَّط أن تضمينات **نموذج الكلمة إلى المتَّجَه** (Word2Vec) تستنبط الارتباطات الدلالية بين الكلمات، كما يتضح من مجموعات الكلمات الواضحة مثل:

- economy (الاقتصاد)، economic (الاقتصادية)، business (الأعمال)، financial (المالية)، sales (المبيعات)، bank (المصرف)، firm (الشركة)، firms (الشركات).
- Internet (الإنترنت)، mobile (الهاتف المحمول)، phones (الهواتف)، phone (الهاتف)، broadband (النطاق العريض)، online (متصل)، digital (رقمي).
- actor (ممثل)، actress (ممثلة)، film (فيلم)، comedy (كوميدي)، films (أفلام)، festival (مهرجان)، band (فرقة)، movie (فيلم).
- game (لعبة)، team (فريق)، match (مباراة)، players (لاعبون)، coach (مدرِّب)، injury (إصابة)، club (نادي)، rugby (الرجبي).


وزارة التعليم
Ministry of Education
2025 - 1447

# البرمجة الاتجاهية للجُمل باستخدام التعلُّم العميق
# Sentence Vectorization with Deep Learning

على الرغم من إمكانية استخدام **نموذج الكلمة إلى المتَّجَه** (Word2Vec) في نمذجة الكلمات الفردية، يتطلب التجميع البرمجة الاتجاهية للنص بأكمله. إحدى الطرائق الأكثر شهرة لتحقيق ذلك هي تمثيلات **ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT) المُستنِدة إلى منهجية التعلُّم العميق.

## تمثيلات الترميز ثنائية الاتجاه من المحولات
## Bidirectional Encoder Representations from Transformers (BERT)

**تمثيلات الترميز ثنائية الاتجاه من المحولات** (BERT) هي نموذج تمثيل لغوي قوي طورته شركة قوقل، ويعدُّ التدريب المُسبَق والضبط الدقيق عاملان رئيسان وراء قدرة **تمثيلات الترميز ثنائية الاتجاه من المحولات** (BERT) على تطبيق نقل التعلُّم، أي القدرة على الاحتفاظ بالمعلومات حول مشكلة ما والاستفادة منها في حلِّ مشكلة أخرى، ويتمّ التدريب المُسبَق عبر تغذية النموذج بكمية هائلة من البيانات غير المُعنوَنة لعدة مهام، مثل التنبؤ اللغوي المُقنَّع (إخفاء الكلمات العشوائية في مُدخَلات النصوص والمُهِمَّة هي التنبؤ بهذه الكلمات). يُهيِّئ نموذج **تمثيلات الترميز ثنائية الاتجاه من المحولات** (BERT) المتغيرات المُدرَّبة مُسبقًا للضبط الدقيق، كما تُستخدَم مجموعات البيانات المُعنوَنة من المهام النهائية لضبط دقة عمل النموذج، ويكون لكل مُهِمَّة نهائية نماذج دقيقة منفصلة، برغم أنها مُهيَّأة بالمتغيرات المُدرَّبة نفسها مُسبقًا. على سبيل المثال، تختلف عملية الضبط الدقيق لنموذج تحليل المشاعر عن نموذج الإجابة على الأسئلة. ومن المهم معرفة أن الفروقات في بنية النماذج تصبح ضئيلة أو منعدمة بعد خطوة ضبط الدقة.

## تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات SBERT

**تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT) هي الإصدار المُعدَّل من **تمثيلات الترميز ثنائية الاتجاه من المحولات** (BERT). تُدرَّب **تمثيلات الترميز ثنائية الاتجاه من المحولات** (BERT) مثل **نموذج الكلمة إلى المتَّجَه** (Word2Vec) للتنبؤ بالكلمات بناءً على سياق الجُمل الواردة بها. ومن ناحية أخرى، تُدرَّب **تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT) للتنبؤ بما إذا كانت جملتان متشابهتين دلاليًا. تُستخدَم **تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT) لإنشاء تضمينات لأجزاء النصوص الأطول من الجُمل، مثل الفقرات، أو النصوص القصيرة، أو المقالات في مجموعة بيانات هيئة الإذاعة البريطانية محل الدراسة في هذه الوحدة. بالرغم من أن النماذج الثلاث تستند جميعها إلى الشبكات العصبية، إلا أن **تمثيلات الترميز ثنائية الاتجاه من المحولات** (BERT) و**تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT) تنفذان بُنية مختلفة بشكل كبير وأكثر تعقيدًا من **نموذج الكلمة إلى المتَّجَه** (Word2Vec).

## مكتبة الجُمل والمحولات Sentence_transformers Library

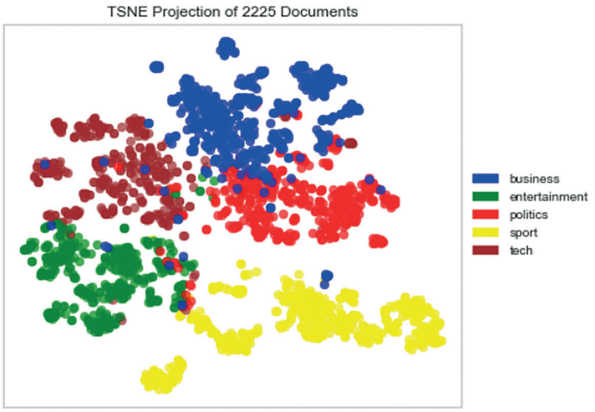
تُطبق **مكتبة الجُمل والمحولات** (sentence_transformers) الوظائف الكاملة لنموذج تمثيلات **ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT). تأتي المكتبة بالعديد من نماذج **تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT) المُدرَّبة مُسبقًا؛ كلٌّ منها مُدرَّب على مجموعة بيانات مختلفة ولتحقيق أهداف مختلفة. يعمل المقطع البرمجي التالي على تحميل أحد النماذج العامة الشهيرة المُدرَّبة مُسبقًا، ويستخدمها لإنشاء تضمينات للمستندات في مجموعة بيانات هيئة الإذاعة البريطانية:

```
%%capture
!pip install sentence_transformers
from sentence_transformers import SentenceTransformer

model = SentenceTransformer('all-MiniLM-L6-v2') # load the pre-trained model.

text_emb = model.encode(bbc_docs) # embed the BBC documents.
```


وزارة التعليم
Ministry of Education
2025 - 1447

لقد استخدمت في وقت سابق في هذه الوحدة أداة تضمين المجاور العشوائي الموزع على شكل T والتي هي (TSNEVisualizer)، لتصوير المُستندات المُمثلة بالمتَّجهات المُنتجة باستخدام أداة تكرار المصطلح-تكرار المستند العكسي (TF-IDF). يمكن الآن استخدامها للتضمينات المُنتَجة بواسطة **تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT):

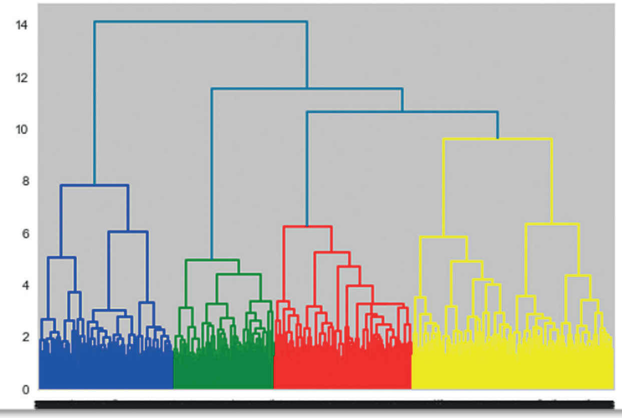
```
tsne = TSNEVisualizer(colors=['blue','green','red','yellow','brown'])
tsne.fit(text_emb,bbc_labels)
tsne.show();
```




شكل 3.23: إسقاط تضمين المجاور العشوائي الموزَّع على شكل T (T-SNE) للتضمينات المُنتجة بواسطة تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات (SBERT)

يوضِّح الشكل أن **تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT) تؤدي إلى فصل أكثر وضوحًا للأقسام الإخبارية المختلفة مع عدد أقل من الشوائب من **تكرار المصطلح-تكرار المُستنَد العكسي** (TF-IDF).
الخطوة التالية هي استخدام التضمينات لتدريب خوارزمية التجميع التكتلي:

```
plt.figure() # create a new figure.

# iteratively merge points and clusters until all points belong to a single cluster. Return the the linkage of the produced tree.
linkage_emb=hierarchy.linkage(text_emb, method='ward')

hierarchy.dendrogram(linkage_emb) # visualize the linkage.
plt.show() # show the figure.
```




شكل 3.24: الرسم الشجري الهرمي لتمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات (SBERT)

وزارة التعليم
Ministry of Education
2025 - 1447

كما هو موضّح في الشكل 3.24، فإن أداة الرسم الشجري تشير إلى 4 عناقيد، كل واحد منها مُميز بلون مختلف. يَستخدِم المقطع البرمجي التالي هذا المقترح لحساب العناقيد وحساب مقاييس التقييم:

```
AC_emb=AgglomerativeClustering(linkage='ward',n_clusters=4)
AC_emb.fit(text_emb)
pred_emb=AC_emb.labels_

print('\nHomogeneity score:',homogeneity_score(bbc_labels,pred_emb))
print('\nAdjusted Rand score:',adjusted_rand_score(bbc_labels,pred_emb))
print('\nCompleteness score:',completeness_score(bbc_labels,pred_emb))
```

```
Homogeneity score: 0.6741395570357063

Adjusted Rand score: 0.6919474005627763

Completeness score: 0.7965514907905805
```

إذا كانت البيانات قد تم إعادة تجميعها باستخدام العدد الصحيح من 5 عناقيد، فالعنقود الأصفر المُحدد بالشكل أعلاه سينقسم إلى اثنين، وستكون النتائج على النحو التالي:

```
AC_emb=AgglomerativeClustering(linkage='ward',n_clusters=5)
AC_emb.fit(text_emb)
pred_emb=AC_emb.labels_

print('\nHomogeneity score:',homogeneity_score(bbc_labels,pred_emb))
print('\nAdjusted Rand score:',adjusted_rand_score(bbc_labels,pred_emb))
print('\nCompleteness score:',completeness_score(bbc_labels,pred_emb))
```

```
Homogeneity score: 0.7865655030556284

Adjusted Rand score: 0.8197670431956582

Completeness score: 0.7887580797775077
```

تُظهِر النتائج أن استخدام **تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT) في البرمجة الاتجاهية للنصوص يَنتج عنه نتائج تجميع مُحسَّنة بالمقارنة مع **تكرار المصطلح-تكرار المُستنَد العكسي** (TF-IDF). إذا كان عدد العناقيد هو 5 **لتكرار المصطلح-تكرار المُستنَد العكسي** (TF-IDF) (القيمة الصحيحة) و4 عناقيد **لتمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT)، فإن المقاييس الثلاثة **لتمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT) لا تزال هي الأعلى بفارق كبير. ثم تزداد الفجوة إذا كان العدد 5 لكلٍ من الطريقتين.

وهذا يُعدُّ دليلًا على إمكانات الشبكات العصبية، التي تسمح لها بُنيتها المتطورة بفهم الأنماط الدلالية المُعقدة في البيانات النصيّة.


وزارة التعليم
Ministry of Education
2025 - 1447

# تمرينات

**1**

| حدِّد الجملة الصحيحة والجملة الخاطئة فيما يلي: | صحيحة | خاطئة |
|---|---|---|
| 1. في التعلُّم غير الموجَّه تُستخدم مجموعات البيانات المُعنوَنة لتدريب النموذج. | ○ | ○ |
| 2. يتطلب التعلُّم غير الموجَّه البرمجة الاتجاهية للبيانات. | ○ | ○ |
| 3. تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات (SBERT) تُعدُّ أفضل من تكرار المصطلح-تكرار المستند العكسي (TF-IDF) للبرمجة الاتجاهية للكلمات. | ○ | ○ |
| 4. يَتبع التجميع التكتلي منهجية التصميم من أعلى إلى أسفل لتحديد العناقيد. | ○ | ○ |
| 5. تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات (SBERT) مُدرَّبة للتنبؤ بما إذا كانت جملتان مختلفتين دلاليًا. | ○ | ○ |

**2** استعرِض بعض التطبيقات التي يُستخدم فيها تقليص الأبعاد، وصِف التقنيات المُستخدَمة فيه.

**3** اشرح وظائف البرمجة الاتجاهية لمقياس تكرار المصطلح-تكرار المستند العكسي (TF-IDF).

**4** لديك مصفوفة numPy تدعى 'Docs' تتضمن مستندًا نصيًا واحدًا في كل صف. لديك كذلك مصفوفة labels تتضمن قيم كل مستند في Docs. أكمل المقطع البرمجي التالي بحيث تستخدِم نموذج تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات (SBERT) المُدرَّب مُسبقًا لحساب تضمينات كل الوثائق في Docs، ثم استخدم أداة TSNEVisualizer تضمين المجاور العشوائي الموزَّع على شكل T لتصوير التضمينات في الفضاء ثنائي الأبعاد، باستخدام لون مختلف لكل واحد من القيم الأربعة المحتملة:

```
from sentence_transformers import ____________________

from ________________ import TSNEVisualizer model =__________________('all-MiniLM-L6-v2') # loads the pre-trained model.

docs_emb = model.________________(Docs) # embeds the docs

tsne = ____________________(__________________=['blue','green','red','yellow'])

tsne.________________(__________________,__________________)

tsne.show();
```

**5** أكمل المقطع البرمجي التالي بحيث تَستخدِم نموذج الكلمة إلى المتَّجَه (Word2Vec) لاستبدال كل كلمة في إحدى الجُمل بأخرى تكون أكثر شبهًا بها:

```
import gensim.downloader as __________________
import re

model_wv = ________________.__________________('word2vec-google-news-300')
old_sentence='My name is John and I like basketball.'
new_sentence=''

for word in re.__________________(r'\b\w\w+\b',old_sentence.lower()):

       replacement=model_wv.______________(positive=['apple'], ______________=1)[0]

       new_sentence+=____________________

sentence=new_sentence.strip()
```

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الثالث

# توليد النص

## توليد اللغات الطبيعية (NLG) Natural Language Generation

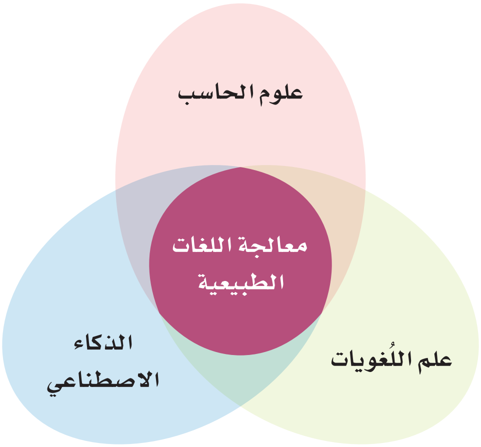
**توليد اللغات الطبيعية** (NLG) هو أحد فروع **معالجة اللغات الطبيعية** (NLP) التي تركّز على توليد النصوص البشرية باستخدام خوارزميات الحاسب. الهدف من **توليد اللغات الطبيعية** (NLG) هو توليد اللغات المكتوبة أو المنطوقة بصورة طبيعية ومفهومة للبشر دون الحاجة إلى تدخل بشري. توجد العديد من المنهجيات المختلفة لتوليد اللغات الطبيعية مثل: المنهجيات المُستنِدة إلى القوالب، والمُستنِدة إلى القواعد، والمُستنِدة إلى تعلُّم الآلة.

**معالجة اللغات الطبيعية (Natural Language Processing-NLP):**

**معالجة اللغات الطبيعية** (NLP) هو أحد فروع الذكاء الاصطناعي الذي يمنح أجهزة الحاسب القدرة على محاكاة اللغات البشرية الطبيعية.

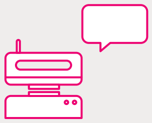
**توليد اللغات الطبيعية (Natural Language Generation-NLG):**

**توليد اللغات الطبيعية** (NLG) هي عملية توليد النصوص البشرية باستخدام **الذكاء الاصطناعي** (AI).




شكل 3.25: مُخطَّط فِنّ (Venn) لمعالجة اللغات الطبيعية (NLP)

**جدول 3.4: تأثير توليد اللغات الطبيعية**

| | |
|---|---|
| | يُستخدم **توليد اللغات الطبيعية** (NLG) لتوليد المقالات والتقارير الإخبارية، والمحتوى المكتوب آليًا مما يوفّر الوقت، ويساعد الأشخاص في التركيز على المهام الإبداعية أو المهام عالية المستوى. |
|  | يمكن الاستفادة من ذلك في تحسين كفاءة وفعالية روبوت الدردشة لخدمة العملاء وتمكينه من تقديم ردود طبيعية ومفيدة لأسئلتهم واستفساراتهم. |
| | يمكن الاستفادة من **توليد اللغات الطبيعية** (NLG) في تحسين إمكانية الوصول لذوي الإعاقة أو لذوي الحواجز اللغوية، بتمكينهم من التواصل مع الآلات بطريقة طبيعية وبديهية تناسبهم. |

وزارة التعليم
Ministry of Education
2025 - 1447

هناك أربع أنواع من **توليد اللغات الطبيعية** (NLG):

## توليد اللغات الطبيعية المبني على القوالب Template-Based NLG

يتضمن توليد اللغات الطبيعية المبنيّ على القوالب استخدام قوالب مُحدَّدة مُسبقًا تحدد بنية ومحتوى النص المتولِّد. تُزَوّد هذه القوالب بمعلومات مُحدَّدة لتوليد النص النهائي. تُعدُّ هذه المنهجية بسيطة نسبيًا وتحقق فعالية في توليد النصوص للمهام المُحدَّدة والمُعرَّفة جيدًا. من ناحية أخرى، قد تواجه صعوبة مع المهام المفتوحة أو المهام التي تتطلب درجة عالية من التباين في النص المُوَلَّد. على سبيل المثال، قالب تقرير حالة الطقس ربما يبدو كما يلي: Today in **[city]**, it is **[temperature]** degrees with **[weather condition]**. (اليوم في **[المدينة]**، درجة الحرارة هي **[درجة الحرارة]** مئوية و**[حالة الطقس]**.).

## توليد اللغات الطبيعية المبني على الاختيار Selection-Based NLG

يتضمن توليد اللغات الطبيعية المبنيّ على الاختيار تحديد مجموعة فرعية من الجُمل أو الفقرات لإنشاء ملخّص للنصّ الأصلي الأكبر حجمًا. بالرغم من أن هذه المنهجية لا تُولِّد نصوصًا جديدة، إلا أنها مُطبقَّة عمليًا على نطاق واسع؛ وذلك لأنّها تأخذ العينات من مجموعة من الجُمل المكتوبة بواسطة البشر، يمكن الحد من مخاطرة توليد النصوص غير المُتنبأ بها أو ضعيفة البنية. على سبيل المثال، مُولِّد تقرير الطقس المبنيّ على الاختيار قد يضم قاعدة بيانات من العبارات مثل: It is hot outside (الطقس حار بالخارج)، و The temperature is rising (درجة الحرارة ترتفع)، و Expect sunny skies (تنبؤات بطقس مُشمس).

## توليد اللغات الطبيعية المبني على القواعد Rule-Based NLG

يَستخدِم توليد اللغات الطبيعية المبنيّ على القواعد مجموعة من القواعد المُحدَّدة مُسبقًا لتوليد النص. قد تحدِّد هذه القواعد طريقة تجميع الكلمات والعبارات لتشكيل الجُمل، أو كيفية اختيار الكلمات وفقًا للسياق المُستخدَمة فيه. عادةً تُستخدم هذه القواعد لتصميم روبوت الدردشة لخدمة العملاء. قد يكون من السهل تطبيق الأنظمة المبنية على القواعد. وفي بعض الأحيان قد تتسم بالجمود ولا تُولِّد مُخرَجات تبدو طبيعية.

## توليد اللغات الطبيعية المبني على تعلُّم الآلة Machine Learning-Based NLG

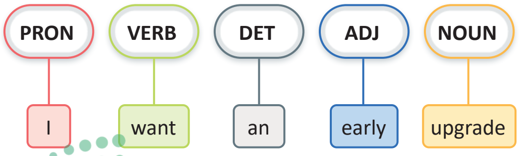
يتضمن توليد اللغات الطبيعية المبنيّ على تعلُّم الآلة تدريب نموذج تعلُّم الآلة على مجموعة كبيرة من بيانات النصوص البشرية. يتعلّم النموذج أنماط النصّ وبنيته، ومن ثَمّ يمكنه توليد النص الجديد الذي يشبه النص البشري في الأسلوب والمحتوى. قد تكون المنهجية أكثر فعالية في المهام التي تتطلب درجة عالية من التباين في النص المُوَلَّد. وقد تتطلب المنهجية مجموعات أكبر من بيانات التدريب والموارد الحسابية.

# استخدام توليد اللغات الطبيعية المبني على القوالب Using Template-Based NLG

توليد اللغات الطبيعية المبنيّ على القوالب بسيط نسبيًا وقد يكون فعالًا في توليد النصوص للمهام المُحدَّدة والمُعرَّفة مثل إنشاء التقارير أو توصيف البيانات. إحدى مميزات توليد اللغات الطبيعية المبنيّ على القوالب هو سهولة التطبيق والصيانة. يُصمِّم الأشخاص القوالب، دون الحاجة إلى خوارزميات تعلُّم الآلة المُعقدَة أو مجموعات كبيرة من بيانات التدريب. وهذا يجعل توليد اللغات الطبيعية المبنيّ على القوالب هو الخيار المناسب للمهام التي تكون ذات بنية ومحتوى نصّ محدّدين، دون الحاجة إلى إجراء تغييرات كبيرة. تَستند قوالب **توليد اللغات الطبيعية** (NLG) إلى أي بُنية لغوية مُحدَّدة مُسبقًا. إحدى الممارسات الشائعة هي إنشاء القوالب التي تتطلَب كلمات بوسوم محددة كجزء من الكلام لإدراجها في الفراغات المُحدَّدة ضمن الجملة.

### وسوم أقسام الكلام Part of Speech (POS) Tags

وسوم **أقسام الكلام** (Part Of Speech)، التي تُعرَّف كذلك باسم وسوم POS هي قيم تُخصَّص للكلمات في النص للإشارة إلى البناء النحوي للكلمات، أو جزء الكلام في الجملة. على سبيل المثال، قد تكون الكلمة اسمًا أو فعلًا أو صفةً أو ظرفًا، إلخ، وتُستخدَم وسوم أقسام الكلام في **معالجة اللغات الطبيعية** (NLP) لتحليل بنية النصّ وفهم معناه.



شكل 3.26: مثال على عملية وسم أقسام الكلام

## تحليل بِناء الجُمل Syntax Analysis

يُستخدم تحليل بِناء الجُمل عادةً إلى جانب وسوم **أقسام الكلام** (POS) في توليد اللغات الطبيعية المبنيّ على القوالب لضمان قدرة القوالب على توليد النصوص الواقعية. يتضمن تحليل بِناء الجُمل التعرّف على أجزاء الكلام في الجُمل، والعلاقات بينها لتحديد البناء النحوي للجُملة. تتضمن الجُملة أنواعًا مختلفة من عناصر بناء الجُملة، مثل:

- **الفعل** (Predicate) هو قسم الجُملة الذي يحتوي على الفعل. وهو عادةً يعبر عمّا يقوم به الفاعل أو عمّا يحدث.
- **الفاعل** (Subject) هو قسم الجُملة الذي يُنفّذ الفعل.
- **المفعول به** (Direct Object) هو اسم أو ضمير يشير إلى الشخص أو الشيء الذي يتأثر مباشرةً بالفعل.

يَستخدِم المقطع البرمجي التالي مكتبة ووندرووردز (Wonderwords) التي تتبع منهجية بِناء الجُمل لعرض بعض الأمثلة على توليد اللغات الطبيعية المبنيّ على القوالب.

```
%%capture

!pip install wonderwords
# used to generate template-based randomized sentences
from wonderwords.random_sentence import RandomSentence

# make a new generator with specific words
generator=RandomSentence(
                    # specify some nouns
                    nouns=["lion", "rabbit", "horse","table"],
                    verbs=["eat","run","laugh"], # specify some verbs.
                    adjectives=['angry','small']) # specify some adjectives.

# generates a sentence with the following template: [subject (noun)] [predicate (verb)]
generator.bare_bone_sentence()
```

```
'The table runs.'
```

```
# generates a sentence with the following template:
# the [(adjective)] [subject (noun)] [predicate (verb)] [direct object (noun)]
generator.sentence()
```

```
'The small lion runs rabbit.'
```

توضح الأمثلة بالأعلى أنه، بينما يُستخدَم توليد اللغات الطبيعية المبنيّ على القوالب لتوليد الجُمل وفق بُنية مُحدَّدة ومُعتمدة مُسبقًا، إلا أنّ هذه الجُمل قد لا تكون ذات مغزىً عملي. وعلى الرغم من إمكانية تحسين دقة النتائج إلى حدٍ كبير بتحديد قوالب متطورة ووضع المزيد من القيود على استخدام المفردات، إلا أن هذه المنهجية غير عملية لتوليد النصوص الواقعية على نطاقٍ واسع. فبدلًا من إنشاء القوالب المُحدَّدة مُسبقًا، تُستخدَم المنهجية الأخرى لتوليد اللغات الطبيعية القائمة على القوالب البنيةَ والمفرداتِ نفسها المُكوِّنة لأي جملة حقيقية كقالب ديناميكي متغير. تتبنى دالة paraphrase() هذه المنهجية.

وزارة التعليم
Ministry of Education
2025 - 1447

## دالة Paraphrase()

تُقسِّم الدالة في البداية النص المُكوَّن من فقرة إلى مجموعة من الجُمل. ثم تحاول استبدال كل كلمة في الجُملة بكلمة أخرى متشابهة دلاليًا. يُقيِّم التشابه الدلالي بواسطة **نموذج الكلمة إلى المتَّجه** (Word2Vec) الذي درسته في الدرس السابق. قد يوصي نموذج الكلمة إلى المتَّجه (Word2Vec) باستبدال الكلمة في الجملة بكلمة أخرى مشابهة لها، مثل: استبدال apple (تفاحة) بـ apples (تفاح)، ولتجنب مثل هذه الحالات تُستخدم دالة مكتبة fuzzywuzzy الشهيرة لتقييم تشابه المُفردات بين الكلمة الأصلية والكلمة البديلة.

الدالة نفسها مُوضحَّة بالأسفل:

```
def paraphrase(text:str, # text to be paraphrased
               stop:set, # set of stopwords
               model_wv,# Word2Vec Model
               lexical_sim_ubound:float, # upper bound on lexical similarity
               semantic_sim_lbound:float # lower bound on semantic similarity
              ):

    words=word_tokenize(text) # tokenizes the text to words

    new_words=[] # new words that will replace the old ones.

    for word in words: # for every word in the text

        word_l=word.lower() # lower-case the word.

        # if the word is a stopword or is not included in the Word2Vec model, do not try to replace it.
        if word_l in stop or word_l not in model_wv:
            new_words.append(word) # append the original word

        else: # otherwise

            # get the 10 most similar words, as per the Word2Vec model.
            # returned words are sorted from most to least similar to the original.
            # semantic similarity is always between 0 and 1.
            replacement_words=model_wv.most_similar(positive=[word_l],
topn=10)

            # for each candidate replacement word
            for rword, sem_sim in replacement_words:
                # get the lexical similarity between the candidate and the original word.
                # the partial_ratio function returns values between 0 and 100.
                # it compares the shorter of the two words with all equal-sized substrings
                # of the original word.
                lex_sim=fuzz.partial_ratio(word_l,rword)

                # if the lexical sim is less than the bound, stop and use this candidate.
                if lex_sim<lexical_sim_ubound:
                    break
```

fuzz تشير إلى مكتبة fuzzywuzzy



وزارة التعليم
Ministry of Education
2025 - 1447

```
            # quality check: if the chosen candidate is not semantically similar enough to
            # the original, then just use the original word.
            if sem_sim<semantic_sim_lbound:
                new_words.append(word)
            else: # use the candidate.
                new_words.append(rword)

    return ' '.join(new_words) # re-join the new words into a single string and return.
```

المُخرَج هو إصدار مُعاد صياغته من النص المُدخَل.

يُستخدَم المقطع البرمجي التالي لاستيراد كل الأدوات اللازمة لدعم دالة paraphrase() وفي المربع الأبيض أدناه، تحصل على مُخرَج طريقة **إعادة الصياغة** (Paraphrase) للنص المُسند إلى المتغير text:

```
%%capture

import gensim.downloader as api # used to download and load a pre-trained Word2Vec model
model_wv = api.load('word2vec-google-news-300')

import nltk
# used to split a piece of text into words. Maintains punctuations as separate tokens
from nltk import word_tokenize
nltk.download('stopwords') # downloads the stopwords tool of the nltk library
# used to get list of very common words in different languages
from nltk.corpus import stopwords
stop=set(stopwords.words('english')) # gets the list of english stopwords

!pip install fuzzywuzzy[speedup]
from fuzzywuzzy import fuzz

text='We had dinner at this restaurant yesterday. It is very close to my
house. All my friends were there, we had a great time. The location is
excellent and the steaks were delicious. I will definitely return soon, highly
recommended!'
# parameters: target text, stopwords, Word2Vec model, upper bound on lexical similarity, lower bound
on semantic similarity
paraphrase(text, stop, model_wv, 80, 0.5)
```

```
'We had brunch at this eatery Monday. It is very close to my bungalow. All
my acquaintances were there, we had a terrific day. The locale is terrific
and the tenderloin were delicious. I will certainly rejoin quickly, hugely
advised!'
```

كما في المنهجيات الأخرى المُستنِدة إلى القوالب، يمكن تحسين النتائج بإضافة المزيد من القيود لتصحيح بعض البدائل الأقل وضوحًا والمذكورة في الأعلى. ومع ذلك، يوضِّح المثال أعلاه أنه يُمكن باستخدام هذه الدالة البسيطة توليد نصوص واقعية.

# استخدام توليد اللغات الطبيعية المبني على الاختيار

## Using Selection-Based NLG

في هذا القسم، ستستعرض منهجية عملية لاختيار نموذج من الجُمل الفرعية من وثيقة مُحدَّدة. هذه المنهجية تُجسِد استخدام ومزايا توليد اللغات الطبيعية المبنيّ على الاختيار يستند إلى لبنتين رئيستين:

- **نموذج الكلمة إلى المتَّجَه** (Word2Vec) المُستخدَم لتحديد أزواج الكلمات المتشابهة دلاليًا.
- مكتبة Networkx الشهيرة ضمن لغة البايثون المُستخدَمة لإنشاء ومعالجة أنواع مختلفة من بيانات الشبكة.

النَّص المُدخَل الذي سيُستخدم في هذا الفصل هو مقالة إخبارية نُشرت بعد المباراة النهائية لكأس العالم 2022.

```
# reads the input document that we want to summarize
with open('article.txt',encoding='utf8',errors='ignore') as f: text=f.read()

text[:100] # shows the first 100 characters of the article
```

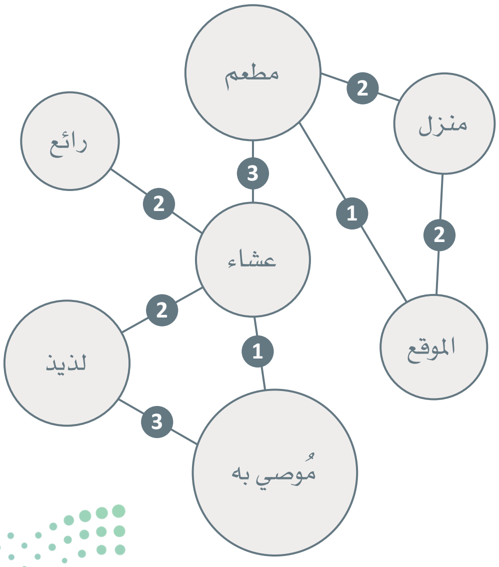
```
'It was a consecration, the spiritual overtones entirely appropriate.
Lionel Messi not only emulated '
```

في البداية، يُرمَّز النص باستخدام مكتبة re والتعبير النمطي نفسه المُستخدَم في الوحدات السابقة:

```
import re # used for regular expressions

# tokenize the document, ignore stopwords, focus only on words included in the Word2Vec model.
tokenized_doc=[word for word in re.findall(r'\b\w\w+\b',text.lower()) if word
not in stop and word in model_wv]

# get the vocabulary (set of unique words).
vocab=set(tokenized_doc)
```

### مكتبة Networkx

يمكن الآن نمذجة مفردات المُستنَد في **مُخطَّط موزون** (Weighted Graph). تُوفر مكتبة Networkx في لغة البايثون مجموعة واسعة من الأدوات لإنشاء وتحليل المُخطَّطات. في توليد اللغات الطبيعية المبنيّ على الاختيار، يُساعد تمثيل مفردات الوثيقة في مُخطَّط موزون في تحديد العلاقات بين الكلمات وتسهيل اختيار العبارات والجُمل ذات الصلة. في المُخطَّط الموزون، تُمثل كل عُقدة كلمةً أو مفهومًا، وتُمثل الحواف بين العُقد العلاقات بين هذه المفاهيم. تُعبر الأوزان على الحواف عن قوة هذه العلاقات، مما يسمح لنظام توليد اللغات الطبيعية بتحديد المفاهيم الأقوى ارتباطًا. عند توليد النصوص، يُستخدم المُخطَّط الموزون للبحث عن العبارات والجُمل استنادًا إلى العلاقات بين الكلمات. على سبيل المثال، قد يَستخدِم النظام المُخطَّط للبحث عن الكلمات والعبارات الأكثر ارتباطًا لوصف كيان مُحدَّد ثم استخدام هذه الكلمات لتحديد الجُملة الأكثر ملاءمةً من قاعدة بيانات النظام.




شكل 3.27: مثال على مُخطَّط موزون لـ Networkx



وزارة التعليم
Ministry of Education
2025 - 1447

## دالة Build_graph( )

تُستخدم دالة Build_graph( ) مكتبة NetworkX لإنشاء مُخطّط يتضمن:

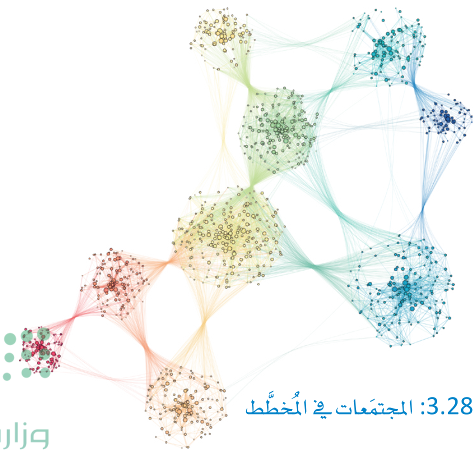
- عُقدة واحدة لكل كلمة ضمن مفردات محددة.
- حافة بين كل كلمتين. الوزن على الحافة يساوي التشابه الدلالي بين الكلمات، المحسوب بواسطة أداة Doc2Vec وهي أداة معالجة اللغات الطبيعية المُخصصة لتمثيل النصّ كمتّجَه وهي تعميم لمنهجية **نموذج الكلمة إلى المتَّجَه** (Word2Vec).

تَرسم الدالة مخطّطًا ذا عُقدة واحدة لكل كلمة في المفردات المُحدَّدة. توجد كذلك حافة بين عُقدتين إذا كان تشابه **نموذج الكلمة إلى المتَّجَه** (Word2Vec) أكبر من الحد المُعطى.

```
# tool used to create combinations (e.g. pairs, triplets) of the elements in a list
from itertools import combinations
import networkx as nx  # python library for processing graphs

def build_graph(vocab:set,  # set of unique words
                model_wv  # Word2Vec model
               ):
    # gets all possible pairs of words in the doc
    pairs=combinations(vocab,2)

    G=nx.Graph() # makes a new graph

    for w1,w2 in pairs:  # for every pair of words w1, w2
        sim=model_wv.similarity(w1, w2) # gets the similarity between the two words
        G.add_edge(w1,w2,weight=sim)

    return G

# creates a graph for the vocabulary of the World Cup document
G=build_graph(vocab,model_wv)
# prints the weight of the edge (semantic similarity) between the two words
G['referee']['goalkeeper']
```

```
{'weight': 0.40646762}
```

وبالنظر إلى ذلك المُخطَّط المبني على الكلمة، يمكن تمثيل مجموعة من الكلمات المتشابهة دلاليًا في صورة عناقيد من العُقد المتصلة معًا بواسطة حواف عالية الوزن. يُطلق على عناقيد العُقد كذلك **المجتمَعات** (Communities). مُخرَج المُخطَّط هو مجموعة بسيطة من الرؤوس والحواف الموزونة. لم تُجرى عملية التجميع حتى الآن لإنشاء المجتمَعات. في الشكل 3.28 تُستخدم ألوان مختلفة لتمييز المجتمَعات في المُخطَّط المذكور بالمثال السابق.



شكل 3.28: المجتمَعات في المُخطَّط

وزارة التعليم
Ministry of Education
2025 - 1447

## خوارزمية لوفان Louvain Algorithm

تتضمن مكتبة Networkx العديد من الخوارزميات لتحليل المُخطَّط والبحث عن المجتمَعات. وواحدة من الخيارات الأكثر فعالية هي خوارزمية لوفان التي تعمل عبر تحريك العُقد بين المجتمَعات حتى تجد بُنية المجتمع التي تمثل الربط الأفضل في الشبكة الضمنية.

### *fx* دالة Get_communities( )

تَستخدم الدالة الآتية خوارزمية لوفان للبحث عن المجتمَعات في المُخطَّط المبنيّ على الكلمات. تَحسب الدالة كذلك مؤشر الأهمية لكل مجتمع على حده، ثم تكون المُخرَجات في صورة قاموسين:

- word_to_community الذي يربط الكلمة بالمجتمع.
- community_scores الذي يربط المجتمع بدرجة الأهمية.

الدرجة تساوي مجموع تكرار الكلمات في المجتمع. على سبيل المثال، إذا كان المجتمع يتضمن ثلاثة كلمات تظهر 5 و6 و8 مرات في النصّ، فإنّ مؤشّر المجتمع حينئذٍ يساوي 19. ومن ناحية المفهوم، يمثل المؤشر جزءًا من النصّ الذي يضُمُّه المجتمع.

```
from networkx.algorithms.community import louvain_communities
from collections import Counter # used to count the frequency of elements in a list

def get_communities( G, # the input graph
                     tokenized_doc:list): # the list of words in a tokenized document

     # gets the communities in the graph
    communities=louvain_communities(G, weight='weight')
    word_cnt=Counter(tokenized_doc)# counts the frequency of each word in the doc

    word_to_community={}# maps each word to its community

    community_scores={}# maps each community to a frequency score

    for comm in communities: # for each community
         # convert it from a set to a tuple so that it can be used as a dictionary key.
        comm=tuple(comm)
        score=0 # initialize the community score to 0.

        for word in comm: # for each word in the community

            word_to_community[word]=comm # map the word to the community

            score+=word_cnt[word] # add the frequency of the word to the community's score.

        community_scores[comm]=score # map the community to the score.

    return word_to_community, community_scores
```


وزارة التعليم
Ministry of Education
2025 - 1447

```
word_to_community, community_scores = get_communities(G,tokenized_doc)
word_to_community['player'][:10] # prints 10 words from the community of the word 'team'
```

```
('champion',
 'stretch',
 'finished',
 'fifth',
 'playing',
 'scoring',
 'scorer',
 'opening',
 'team',
 'win')
```

الآن بعد ربط كل الكلمات بالمجتمع، وربط المجتمع بمؤشر الأهمية، ستكون الخطوة التالية هي استخدام هذه المعلومات لتقييم أهمية كل جملة في المُستنَد الأصلي. دالة ( )evaluate_sentences مُصمَّمة لهذا الغرض.

## دالة ( )Evaluate_sentences

تبدأ الدالة بتقسيم المُستنَد إلى جُمل، ثم حساب مؤشر الأهمية لكل جُملة، استنادًا إلى الكلمات التي تتضمنها. تكتسب كل كلمة مؤشر الأهمية من المجتمع الذي تنتمي إليه.

على سبيل المثال، لديك جملة مكونة من خمسة كلمات w1، w2، w3، w4، w5. الكلمتان w1 وw2 تنتميان إلى مجتمع بمؤشر قيمته 25، والكلمتان w3 وw4 تنتميان إلى مجتمع بمؤشر قيمته 30، والكلمة w5 تنتمي إلى مجتمع بمؤشر قيمته 15. مجموع مؤشرات الجُمل هو 25+25+30+30+15=125. تَستخدِم الدالة بعد ذلك هذه المؤشرات لتصنيف الجُمل في ترتيب تنازلي، من الأكثر إلى الأقل أهمية.

```
from nltk import sent_tokenize # used to split a document into sentences

def evaluate_sentences(doc:str, # original document
                       word_to_community:dict,# maps each word to its community
                       community_scores:dict, # maps each community to a score
                       model_wv): # Word2Vec model

    # splits the text into sentences
    sentences=sent_tokenize(doc)
    scored_sentences=[]# stores (sentence, score) tuples

    for raw_sent in sentences: # for each sentence

        # get all the words in the sentence, ignore stopwords and focus only on words that are in the
Word2Vec model.
        sentence_words=[word
        for word in re.findall(r'\b\w\w+\b',raw_sent.lower()) # tokenizes
            if word not in stop and # ignores stopwords
```

```
                word in model_wv] # ignores words that are not in the Word2Vec model

        sentence_score=0  # the score of the sentence

        for word in sentence_words: # for each word in the sentence

            word_comm=word_to_community[word] # get the community of this word
            sentence_score+=community_scores[word_comm] # add the score of this
community to the sentence score.

        scored_sentences.append((sentence_score,raw_sent)) # stores this sentence and
its total score

    # scores the sentences by their score, in descending order
    scored_sentences=sorted(scored_sentences,key=lambda x:x[0],reverse=True)

    return scored_sentences


scored_sentences=evaluate_sentences(text,word_to_community,community_
scores,model_wv)
len(scored_sentences)
```

```
61
```

يتضمن المُستنَد الأصلي إجمالي 61 جُملة، ويُستخدَم المقطع البرمجي التالي للعثور على الجُمل الثلاثة الأكثر أهمية من بين هذه الجُمل:

```
for i in range(3):
    print(scored_sentences[i],'\n')
```

```
(3368, 'Lionel Messi not only emulated the deity of Argentinian football,
Diego Maradona, by leading the nation to World Cup glory; he finally
plugged the burning gap on his CV, winning the one title that has eluded
him – at the fifth time of asking, surely the last time.')

(2880, 'He scored twice in 97 seconds to force extra-time; the first a
penalty, the second a sublime side-on volley and there was a point towards
the end of regulation time when he appeared hell-bent on making sure that
the additional period would not be needed.')

(2528, 'It will go down as surely the finest World Cup final of all time,
the most pulsating, one of the greatest games in history because of how
Kylian Mbappé hauled France up off the canvas towards the end of normal
time.')
```


وزارة التعليم
Ministry of Education
2025 - 1447

```
print(scored_sentences[-1]) # prints the last sentence with the lowest score
print()
print(scored_sentences[30]) # prints a sentence at the middle of the scoring scale
```

```
(0, 'By then it was 2-0.')

(882, 'Di María won the opening penalty, exploding away from Ousmane
Dembélé before being caught and Messi did the rest.')
```

النتائج تؤكد أن هذه المنهجية تُحدِّد بنجاح الجُمل الأساسية التي تستنبط النقاط الرئيسة في المُستنَد الأصلي، مع تعيين مؤشرات أقل للجُمل الأقل دلالةً. تُطبَّق المنهجية نفسها كما هي لتوليد ملخّص لأي وثيقة مُحدَّدة.

## استخدام توليد اللغات الطبيعية المبني على القواعد لإنشاء روبوت الدردشة

## Using Rule-Based NLG to Create a Chatbot

في هذا القسم، ستُصمِّم روبوت دردشة (Chatbot) وفق المسار المُحدَّد الموصى به بالجمع بين قواعد المعرِفة الرئيسة للأسئلة والأجوبة والنموذج العصبي **تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT)، ويشير هذا إلى أن نقل التعلُّم المُستخدَم في **تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT) له البنية نفسها كما في **تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT) all-MiniLM-L6-v2 وسوف يهيَّأ بشكل دقيق لمُهِمَّة أخرى غير تحليل المشاعر، وهي: توليد اللغات الطبيعية.

### 1. تحميل نموذج تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات المُدرَّب مُسبقًا

### Load the Pre-Trained SBERT Model

الخطوة الأولى هي تحميل نموذج **تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT) المُدرَّب مُسبقًا:

```
%%capture
from sentence_transformers import SentenceTransformer, util
model_sbert = SentenceTransformer('all-MiniLM-L6-v2')
```

### 2. إنشاء قاعدة معرِفة بسيطة Create a Simple Knowledge Base

الخطوة الثانية هي إنشاء قاعدة معرِفة بسيطة لتحديد النص البرمجي المكون من الأسئلة والأجوبة التي يستخدمها روبوت الدردشة. يتضمن النص البرمجي 4 أسئلة (السؤال 1 إلى 4) والأجوبة على كل سؤال (الإجابة 1 إلى 4). كل إجابة مكونة من مجموعة من الخيارات كل خيار يتكون من قيمتين فقط، تُمثِّل القيمة الثانية السؤال التالي الذي يستخدمه روبوت الدردشة. إذا كان هذا هو السؤال الأخير، ستكون القيمة الثانية خالية. هذه الخيارات تمثل الإجابات الصحيحة المحتملة على الأسئلة المعنية بها. على سبيل المثال، الإجابة على السؤال الثاني لها خياران محتملان ["Java",None] and ["Python",None] (["جافا"،لا يوجد] و ["البايثون"،لا يوجد]). كل خيار مُكون من قيمتين:

- النص الحقيقي للإجابة المقبولة مثل: Java (جافا) أو Courses on Marketing (دورات تدريبية في التسويق).
- مُعرِّف يشير إلى السؤال التالي الذي سيطرحه روبوت الدردشة عند تحديد هذا الخيار. على سبيل المثال، إذا حدَّد المُستخدِم خيار ["Courses on Engineering","3"] (["دورات تدريبية في الهندسة"، "3"]) كإجابة على السؤال الأول، يكون السؤال التالي الذي سيطرحه روبوت الدردشة هو السؤال الثالث.



وزارة التعليم
Ministry of Education
2025 - 1447

يمكن توسيع قاعدة المعرِفة البسيطة لتشمل مستويات أكثر من الأسئلة والأجوبة، وتجعل روبوت الدردشة أكثر ذكاءً.

```
QA={
       "Q1":"What type of courses are you interested in?",
       "A1":[["Courses in Computer Programming","2"],
             ["Courses in Engineering","3"],
             ["Courses in Marketing","4"]],

       "Q2":"What type of Programming Languages are you interested in?",
       "A2":[["Java",None],["Python",None]],

       "Q3":"What type of Engineering are you interested in?",
       "A3":[["Mechanical Engineering",None],["Electrical Engineering",None]],

       "Q4":"What type of Marketing are you interested in?",
       "A4":[["Social Media Marketing",None],["Search Engine
Optimization",None]]
}
```

## دالة Chat()

في النهاية، تُستخدَم دالة Chat() لمعالجة قاعدة المعرِفة وتنفيذ روبوت الدردشة. بعد طرح السؤال، يقرأ روبوت الدردشة رد المُستخدِم.

- إن كان الرد مشابهًا دلاليًا لأحد خيارات الإجابات المقبولة لهذا السؤال، يُحدَّد ذلك الخيار وينتقل روبوت الدردشة إلى السؤال التالي.
- إن لم يتشابه الرد مع أيٍ من الخيارات، يُطلب من المُستخدِم إعادة صياغة الرد.

تُستخدَم دالة **تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT) لتقييم مؤشر التشابه الدلالي بين الرد وكل الخيارات المُرشَّحة. يُعدُّ الخيار متشابهًا إذا كان المؤشر أعلى من مُتغير الحد الأدنى sim_lbound.

```
import numpy as np # used for processing numeric data

def chat(QA:dict, # the Question-Answer script of the chatbot
        model_sbert, # a pre-trained SBERT model
        sim_lbound:float): # lower bound on the similarity between the user's response and the
closest candidate answer

    qa_id='1' # the QA id

    while True: # an infinite loop, will break in specific conditions

        print('>>',QA['Q'+qa_id]) # prints the question for this qa_id
        candidates=QA["A"+qa_id] # gets the candidate answers for this qa_id

        print(flush=True) # used only for formatting purposes
        response=input() # reads the user's response

        # embed the response
        response_embeddings = model_sbert.encode([response], convert_to_
tensor=True)
        # embed each candidate answer. x is the text, y is the qa_id. Only embed x.
```


وزارة التعليم
Ministry of Education
2025 - 1447

```python
        candidate_embeddings = model_sbert.encode([x for x,y in candidates],
convert_to_tensor=True)

        # gets the similarity score for each candidate
        similarity_scores = util.cos_sim(response_embeddings, candidate_
embeddings)

        # finds the index of the closest answer.
        # np.argmax(L) finds the index of the highest number in a list L
        winner_index=np.argmax(similarity_scores[0])

        # if the score of the winner is less than the bound, ask again.
        if similarity_scores[0][winner_index]<sim_lbound:
            print('>> Apologies, I could not understand you. Please rephrase
your response.')
            continue

        # gets the winner (best candidate answer)
        winner=candidates[winner_index]

        # prints the winner's text
        print('\n>> You have selected:',winner[0])
        print()

        qa_id=winner[1] # gets the qa_id for this winner

        if qa_id==None: # no more questions to ask, exit the loop
            print('>> Thank you, I just emailed you a list of courses.')
            break
```

أنظر إلى التفاعلين التاليين بين روبوت الدردشة والمُستخدِم:

**التفاعل الأول**

```python
chat(QA,model_sbert, 0.5)
```

```
>> What type of courses are you interested in?
marketing courses
>> You have selected: Courses on Marketing
>> What type of Marketing are you interested in?
seo
>> You have selected: Search Engine Optimization
>> Thank you, I just emailed you a list of courses.
```

في التفاعل الأول، يفهم روبوت الدردشة أن المُستخدِم يبحث عن دورات تدريبية في التسويق. وكذلك، روبوت الدردشة ذكي بالقدر الكافي ليفهم أن المصطلح SEO يشبه دلاليًا مصطلح Search Engine Optimization (تحسين محركات البحث) مما يؤدي إلى إنهاء المناقشة بنجاح.

التفاعل الثاني

```
chat(QA,model_sbert, 0.5)
```

```
>> What type of courses are you interested in?

cooking classes
>> Apologies, I could not understand you. Please rephrase your response.
>> What type of courses are you interested in?

software courses

>> You have selected: Courses on Computer Programming

>> What type of Programming Languages are you interested in?

C++

>> You have selected: Java

>> Thank you, I just emailed you a list of courses.
```

في التفاعل الثاني، يفهم روبوت الدردشة أن Cooking Classes (دروس الطهي) لا تشبه دلاليًا الخيارات الموجودة في قاعدة المعرفة. وهو ذكي بالقدر الكافي ليفهم أن Software courses (الدورات التدريبية في البرمجة) يجب أن ترتبط بخيار Courses on Computer Programming (الدورات التدريبية في برمجة الحاسب). الجزء الأخير من التفاعل يسلط الضوء على نقاط الضعف: يربط روبوت الدردشة بين رد المُستخدِم ++C و Java. على الرغم من أن لغتي البرمجة مرتبطتان بالفعل ويمكن القول بأنهما أكثر ارتباطًا من لغتي البايثون و ++C، إلا أن الرد المناسب يجب أن يُوضِّح أن روبوت الدردشة لا يتمتع بالدراية الكافية للتوصية بالدورات التدريبية في لغة ++C. إحدى الطرائق لمعالجة هذا القصور هي استخدام التشابه بين المفردات بدلًا من التشابه الدلالي للمقارنة بين الردود والخيارات ذات الصلة ببعض الأسئلة.

# استخدام تعلُّم الآلة لتوليد نص واقعي

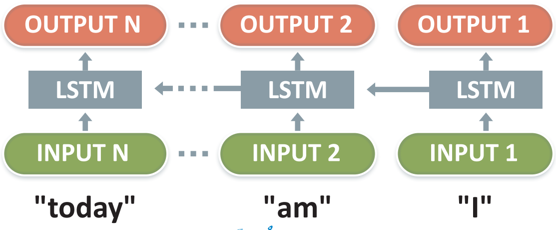
## Using Machine Learning to Generate Realistic Text

الطرائق الموضحة في الأقسام السابقة تَستخدِم القوالب، والقواعد، أو تقنيات التحديد لتوليد النصوص للتطبيقات المختلفة. في هذا القسم، ستتعرَّف على أحدث تقنيات تعلُّم الآلة المُستخدَمة في **توليد اللغات الطبيعية** (NLG).

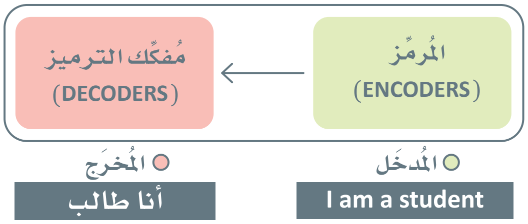
**جدول 3.5: تقنيات تعلُّم الآلة المُتقدمة المُستخدَمة في توليد اللغات الطبيعية**

| التقنية | الوصف |
|---|---|
| **شبكة الذاكرة المُطولّة قصيرة المدى** (Long Short-Term Memory – LSTM) | تتكون شبكة **الذاكرة المُطولّة قصيرة المدى** (LSTM) من خلايا **ذاكرة** (Memory Cells) مرتبطة ببعض. عند إدخال سلسلة من البيانات إلى الشبكة، تتولى معالجة كل عنصر في السلسلة واحدًا تلو الآخر، وتُحدِّث الشبكة خلايا الذاكرة لتوليد مُخرَج لكل عنصر على حده. شبكات **الذاكرة المُطولّة قصيرة المدى** (LSTM) تناسب مهام **توليد اللغات الطبيعية** (NLG) لقدرتها على الاحتفاظ بالمعلومات من سلاسل البيانات (مثل التعرّف على الكلام أو الكتابة اليدوية) ومعالجة تعقيد اللغات الطبيعية. |
| **النماذج المبنية على المحولات** (Transformer-Based Models) | النماذج المبنية على المحولات هي تلك التي تفهم اللغات البشريّة وتولّدها، وتَستنِد هذه النماذج في عملها إلى تقنية **الانتباه الذاتي** (Self-Attention) التي تمكِّنها من فهم العلاقات بين الكلمات المختلفة في الجُمل. |


وزارة التعليم
Ministry of Education
2025 - 1447




شكل 3.29: الذاكرة المُطوّلة قصيرة المدى




شكل 3.30: المُحوِّل

## المُحوِّلات Transformers

المُحوِّلات مناسبة لمهام توليد اللغات الطبيعية لقدرتها على معالجة البيانات المُدخَلة المُتسلسلة بكفاءة. في نموذج المُحوِّلات، تُمرَّر البيانات المُدخَلة عبر المُرمِّز الذي يُحوِّل المُدخَلات إلى تمثيل مستمر، ثم يُمرَّر التمثيل عبر مُفكِّك الترميز الذي يُولِّد التسلسل المُخرَج. إحدى الخصائص الرئيسة لهذه النماذج هي استخدام آليات الانتباه التي تسمح للنموذج بالتركيز على الأجزاء المُهِمَّة من التسلسل في حين تتجاهل الأجزاء الأقل دلالةً. أظهرت نماذج المُحوِّلات كفاءة في توليد النص عالي الدقة للعديد من مهام توليد اللغات الطبيعية بما في ذلك ترجمة الآلة، والتلخيص، والإجابة على الأسئلة.

## نموذج الإصدار الثاني من المُحوِّل التوليدي مُسبَق التدريب GPT-2 Model

في هذا القسم، ستستخدِم **الإصدار الثاني من المُحوِّل التوليدي مُسبَق التدريب** (GPT-2) وهو نموذج لغوي قوي طورته شركة أوبن أي آي (OpenAI) لتوليد النصوص المُستنِدة إلى النص التلقيني المُدخَل بواسطة المُستخدِم. **الإصدار الثاني من المُحوِّل التوليدي مُسبق التدريب** (Generative Pre-training Transformer 2 - GPT-2) مُدرَّب على مجموعة بيانات تضم أكثر من ثمان ملايين صفحة ويب ويتميز بالقدرة على إنشاء النصوص البشرية بعدَّة لغات وأساليب. بُنية **الإصدار الثاني من المُحوِّل التوليدي مُسبق التدريب** (GPT-2) المبنية على المُحوِّل تسمح بتحديد **التبعيَّات** (Dependencies) بعيدة المدى وتوليد النصوص المُتَّسقة، وهو مُدرَّب للتنبؤ بالكلمة التالية وفقًا لكل الكلمات السابقة ضمن النص، وبالتالي، يمكن استخدام النموذج لتوليد نصوص طويلة جدًا عبر التنبؤ المستمر وإضافة المزيد من الكلمات.

```python
%%capture
!pip install transformers
!pip install torch
import torch # an open-source machine learning library for neural networks, required for GPT2.
from transformers import GPT2LMHeadModel, GPT2Tokenizer

# initialize a tokenizer and a generator based on a pre-trained GPT2 model.

# used to:
# -encode the text provided by the user into tokens
# -translate (decode) the output of the generator back to text
tokenizer = GPT2Tokenizer.from_pretrained('gpt2')

# used to generate new tokens based on the inputted text
generator = GPT2LMHeadModel.from_pretrained('gpt2')
```

يُقدَّم النص التالي كأساس يستند إليه **الإصدار الثاني من المُحوِّل التوليدي مُسبق التدريب** (GPT-2):

```python
text='We had dinner at this restaurant yesterday. It is very close to my
house. All my friends were there, we had a great time. The location is
```

وزارة التعليم
Ministry of Education
2025 - 1447

```
excellent and the steaks were delicious. I will definitely return soon, highly
recommended!'

# encodes the given text into tokens
encoded_text = tokenizer.encode(text, return_tensors='pt')

# use the generator to generate more tokens.
# do_sample=True prevents GPT-2 from just predicting the most likely word at every step.
generated_tokens = generator.generate(encoded_text,
                                      max_length=200) # max number of new tokens to
generate
#decode the generates tokens to convert them to words
# skip_special_tokens=True is used to avoid special tokens such as '>' or '-' characters.
print(tokenizer.decode(generated_tokens[0], skip_special_tokens=True))
```

```
We had dinner at this restaurant yesterday. It is very close to my house.
All my friends were there, we had a great time. The location is excellent
and the steaks were delicious.I will definitely return soon, highly
recommended!

I've been coming here for a while now and I've been coming here for a while
now and I've been coming here for a while now and I've been coming here for
a while now and I've been coming here for a while now and I've been coming
here for a while now and I've been coming here for a while now and I've
been coming here for a while now and I've been coming here for a while now
and I've been coming here for a while now and I've been coming here for a
while now and I've been coming here for a while now and I've been coming
here for a while now and I've been coming here for a while now and I've
been coming here for a while now and
```

```
# use the generator to generate more tokens.
# do_sample=True prevents GPT-2 from just predicting the most likely word at every step.
generated_tokens = generator.generate(encoded_text,
                                      max_length=200, # max number of new tokens to
generate
                                      do_sample=True)

print(tokenizer.decode(generated_tokens[0],skip_special_tokens=True))
```

```
We had dinner at this restaurant yesterday. It is very close to my house.
All my friends were there, we had a great time. The location is excellent
and the steaks were delicious.I will definitely return soon, highly
recommended!

If you just found this place helpful. If you like to watch videos or
go to the pool while you're there, go for it! Good service - I'm from
Colorado and love to get in and out of this place. The food was amazing!
Also, we were happy to see the waitstaff with their great hands - I went
for dinner. I ordered a small side salad (with garlic on top), and had a
slice of tuna instead. When I was eating, I was able to get up and eat my
salad while waiting for my friend to pick up the plate, so I had a great
time too. Staff was welcoming and accommodating. Parking is cheap in this
neighborhood, and it is in the neighborhood that it needs to
```


وزارة التعليم
Ministry of Education
2025 - 1447

يحقّق هذا مُخرَجات أكثر تنوعًا، مع الحفاظ على دقة وسلامة النص المولَّد، حيث يستخدِم النص مفردات غنية وهو سليم نحويًا. يسمح **الإصدار الثاني من المُحوِّل التوليدي مُسبق التدريب** (GPT-2) بتخصيص المُخرَج بشكل أفضل. يتضح ذلك عند استخدام مُتغير temperature (درجة الحرارة) الذي يسمح للنموذج بتقبل المزيد من المخاطر بل وأحيانًا اختيار بعض الكلمات الأقل احتمالًا. القيم الأعلى لهذا المُتغير تؤدي إلى نصوص أكثر تنوعًا، مثل:

```
# Generate tokens with higher diversity
generated_tokens = generator.generate(
    encoded_text, max_length=200, do_sample=True, temperature=2.0)

print(tokenizer.decode(generated_tokens[0], skip_special_tokens=True))
```

```
We had dinner at this restaurant yesterday. It is very close to my house.
All my friends were there, we had a great time. The location is excellent
and the steaks were delicious.I will definitely return soon, highly
recommended!

Worth a 5 I thought a steak at a large butcher was the end story!! We were
lucky. The price was cheap!! That night though as soon as dinner was on
my turn that price cut completely out. At the tail area they only have
french fries or kiwifet - no gravy - they get a hard egg the other day too
they call kawif at 3 PM it will be better this summer if I stay more late
with friends. When asked it takes 2 or 3 weeks so far to cook that in this
house. Once I found a place it was great. Everything I am waiting is just
perfect as usual....great prices especially at one where a single bite
would suffice or make more as this only runs on the regular hours
```

ومع ذلك، إذا كانت درجة الحرارة مرتفعة للغاية، فإنّ النموذج سيتجاهل الإرشادات الأساسية التي تظهر في **المُدخَل الأوليّ** (Original Seed) ويُولِّد مُخرَجًا أقل واقعية وليس له معنى:

```
# Too high temperature leads to divergence in the meaning of the tokens
generated_tokens = generator.generate(
    encoded_text, max_length=200, do_sample=True, temperature=4.0)

print(tokenizer.decode(generated_tokens[0], skip_special_tokens=True))
```

```
We had dinner at this restaurant yesterday. It is very close to my house.
All my friends were there, we had a great time. The location is excellent
and the steaks were delicious.I will definitely return soon, highly
recommended! It has the nicest ambagas of '98 that I like; most Mexican.
And really nice steak house; amazing Mexican atmosphere to this very
particular piece of house I just fell away before its due date, no surprise
my 5yo one fell in right last July so it took forever at any number on
it being 6 (with it taking two or sometimes 3 month), I really have found
comfort/affability on many more restaurants when ordering.If you try at
it they tell ya all about 2 and three places will NOT come out before they
close them/curry. Also at home i would leave everything until 1 hour but
sometimes wait two nights waiting for 2+ then when 2 times you leave you
wait in until 6 in such that it works to
```

# تمرينات

**1**

| حدِّد الجملة الصحيحة والجملة الخاطئة فيما يلي: | صحيحة | خاطئة |
|---|---|---|
| 1. توليد اللغات الطبيعية المبنيّ على تعلُّم الآلة يتطلب مجموعات كبيرة من بيانات التدريب والموارد الحسابية. | ● | ● |
| 2. الفعل هو نوع من وسوم أقسام الكلام (POS). | ● | ● |
| 3. في تحليل بِناء الجُمل لتوليد اللغات الطبيعية المبنيّ على القوالب، يُستخدَم التحليل بصورة منفصلة عن وسوم أقسام الكلام (POS). | ● | ● |
| 4. المجتمَعات هي عناقيد العُقد التي تُمثِّل الكلمات المختلفة دلاليًا. | ● | ● |
| 5. يصبح روبوت الدردشة أكثر ذكاءً كلما ازداد عدد مستويات الأسئلة والأجوبة المُضافة إلى قاعدة المعرِفة. | ● | ● |

**2** قارن بين المنهجيات المختلفة لتوليد اللغات الطبيعية (NLG).

**3** حدِّد ثلاث تطبيقات مختلفة لتوليد اللغات الطبيعية (NLG).



وزارة التعليم
Ministry of Education
2025 - 1447

**4** أكمل المقطع البرمجي التالي حتى تقبل الدالة ()build_graph مفردات مُحدَّدة من الكلمات ونموذج الكلمة إلى المتَّجَه (Word2Vec) المُدرَّب لرسم مُخطَّط ذي عُقدة واحدة لكل كلمة في المفردات المُحدَّدة. يجب أن يحتوي المُخطَّط على حافة بين عُقدتين إذا كان تشابه نموذج الكلمة إلى المتَّجَه (Word2Vec) أكبر من مستوى التشابه المُعطى، ويجب ألا تكون هناك أوزان على الحواف.

```
from ______________ import combinations # tool used to create combinations

import networkx as nx # python library for processing graphs

def build_graph(vocab:set, # set of unique words

                model_wv, # Word2Vec model

                similarity_threshold:float

               ):

    pairs=combinations(vocab,_________________) # gets all possible pairs of words in the vocabulary

    G=nx._________________ # makes a new graph

    for w1,w2 in pairs: # for every pair of words w1,w2

        sim=model_wv._________________(w1, w2)# gets the similarity between the two words

        if _______________:

            G._________________(w1,w2)

    return G
```

5 أكمل المقطع البرمجي التالي حتى تَستخدِم الدالة ()get_max_sim نموذج تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات (SBERT) للمقارنة بين جُملة مُحدَّدة my_sentence وكل الجُمل الواردة في قائمة أخرى من الجُمل L. يجب أن تُعيد الدالة الجُملة ذات مُؤشر التشابه الأعلى من L1 إلى my_sentence.

```
from sentence_transformers import ___________________, util

from ______________ import combinations # tool used to create combinations

model_sbert = ___________________ ('all-MiniLM-L6-v2')

def get_max_sim(L1,my_sentence):

    # embeds my_sentence

    my_embedding = model_sbert,________________([my_sentence], convert_to_tensor=True)

    # embeds the sentences from L2

    L_embeddings = model_sbert._______________(L, convert_to_tensor=True)

    similarity_scores = _______________.cos_sim(_________________, ___________)

    winner_index=np.argmax(similarity_scores[0])

    return _______________
```

# المشروع

تصنيف النص هو عملية مكونة من خطوتين تشمل:

**الخطوة الأولى:** استخدام مجموعة من نصوص التدريب ذات القيم (التصنيفات) المعروفة لتدريب نموذج التصنيف.

**الخطوة الثانية:** استخدام نموذج التدريب للتنبؤ بالقيم لكل نصّ في مجموعة بيانات الاختبار. القيم في مجموعة بيانات الاختبار إما غير معروفة أو مخبَّأة وتُستخدم لاحقًا في عملية التحقق.

يجب تمثيل النصوص في كل من مجموعات بيانات التدريب والاختبار بالمتَّجَهات قبل استخدامها. تُستخدَم أدوات CountVectorizer أو TfidfVectorizer من مكتبة **سكليرن** (Sklearn) في البرمجة الاتجاهية.

تُقدِّم مكتبة **سكليرن** (Sklearn) في لغة البايثون قائمة طويلة من نماذج التصنيف. مثل:

- GradientBoostingClassifier( )
- DecisionTreeClassifier( )
- RandomForestClassifier( )

مهمتك هي استخدام مجموعة بيانات التدريب IMDB المُستخدَمة في هذا الدرس لتدريب النموذج الذي يحقق أعلى درجة من الدقة على مجموعة بيانات الاختبار IMDB (imdb_data/imdb_ test.csv). يمكنك تحقيق ذلك عبر:

1. استبدال المُصنِّف MultinomialNB بنماذج تصنيف أخرى من مكتبة **سكليرن** (Sklearn) مثل الموضحة بالأعلى.
2. إعادة تشغيل المفكرة التفاعلية لديك بعد الاستبدال، لحساب دقة كل نموذج جديد بعد تجربته.
3. إنشاء تقرير للمقارنة بين دقة كل النماذج التي جرَّبتها وتحديد النموذج الذي حقق نتائجَ دقيقة.

وزارة التعليم
Ministry of Education
2025 - 1447

# ماذا تعلّمت

- تصنيف النص باستخدام نماذج التعلُّم غير الموجَّه.
- تحليل النص باستخدام نماذج التعلُّم الموجَّه.
- استخدام نماذج تعلُّم الآلة لتوليد اللغات الطبيعية.
- برمجة روبوت دردشة بسيط.

## المصطلحات الرئيسة

| | |
|---|---|
| Part of Speech (POS) Tags | وسوم أقسام الكلام |
| Sentiment Analysis | تحليل المشاعر |
| Supervised Learning | التعلُّم الموجَّه |
| Syntax Analysis | تحليل بِناء الجُمل |
| Tokenization | التقسيم |
| Transfer Learning | التعلُّم المنقول |
| Unsupervised Learning | التعلُّم غير الموجَّه |
| Vectorization | البرمجة الاتجاهية |

| | |
|---|---|
| Black-Box predictors | مُتنبِّئات الصندوق الأسود |
| Chatbot | روبوت الدردشة |
| Cluster | عنقود |
| Dendrogram | الرسم الشجري |
| Dimensionality Reduction | تقليص الأبعاد |
| Document Clustering | تجميع المُستنَدات |
| Natural Language Generation | توليد اللغات الطبيعية |
| Natural Language Processing | معالجة اللغات الطبيعية |

وزارة التعليم
Ministry of Education
2025 - 1447

# الجزء الثاني

## الوحدة الرابعة
التعرّف على الصور

## الوحدة الخامسة
خوارزميات التحسين واتخاذ القرار

## الوحدة السادسة
الذكاء الاصطناعي والمجتمع

وزارة التعليم
Ministry of Education
2025 - 1447

وزارة التعليم
Ministry of Education
2025 - 1447

# 4. التعرّف على الصور

سيتعرّف الطالب في هذه الوحدة على التعلُّم الموجَّه وغير الموجَّه، وكيفية توظيفهما للتعرّف على الصور (Image Recognition) عن طريق إنشاء نموذج وتدريبه؛ ليصبح قادرًا على تصنيف صور لرؤوس الحيوانات أو تجميعها. وسيتعرّف أيضًا على توليد الصور (Image Generation) وكيفية تغييرها، أو إكمال الأجزاء الناقصة فيها مع الحفاظ على واقعيتها.

## أهداف التعلُّم

بنهاية هذه الوحدة سيكون الطالب قادرًا على أن:

- يُعالج الصور معالجة أولية ويستخلص خصائصها.
- يُدرِّب نموذج تعلُّم موجَّه خاص بتصنيف الصور.
- يُعرِّف هيكل الشبكة العصبية.
- يُدرِّب نموذج تعلُّم غير موجَّه خاص بتجميع الصور.
- يُولِّد صورًا بناءً على توجيه نصّي.
- يُكمل الأجزاء الناقصة في صورة مُعطاة بطريقة واقعية.

## الأدوات

- مفكّرة جوبيتر (Jupyter Notebook)
- قوقل كولاب (Google Colab)

وزارة التعليم
Ministry of Education
2025 - 1447

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الأول

# التعلُّم الموجَّه لتحليل الصور

## التعلُّم الموجَّه في رؤية الحاسب Supervised Learning for Computer Vision

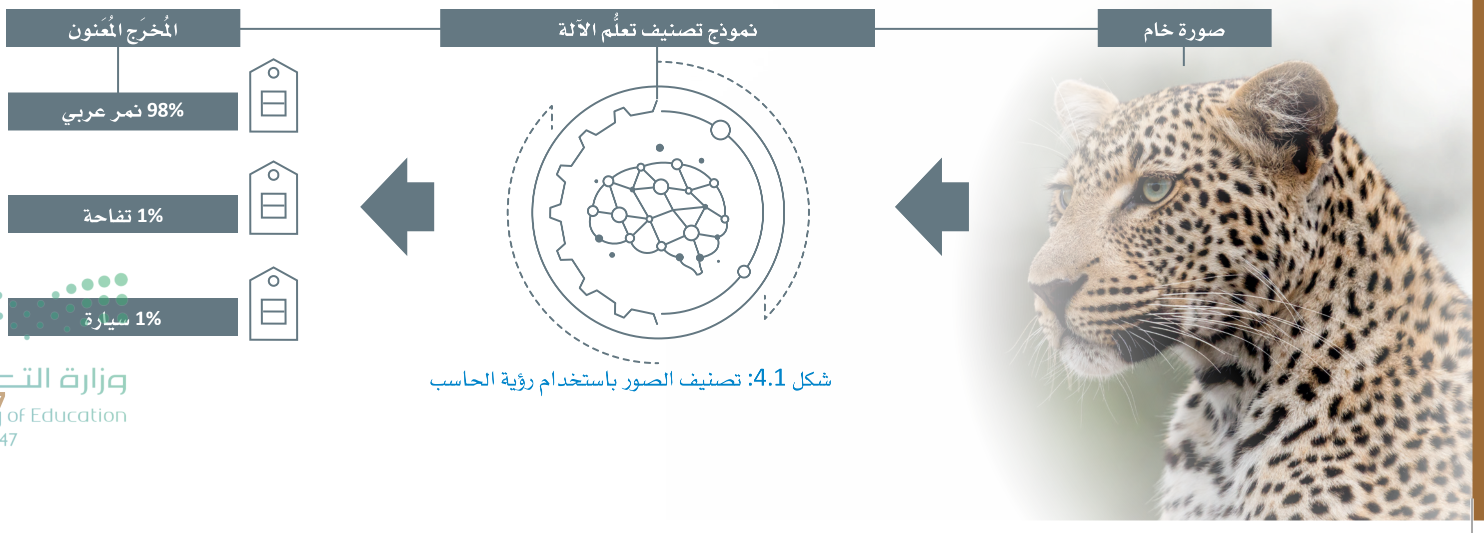
تُعدُّ **رؤية الحاسب** (Computer Vision) مجالًا فرعيًا من مجالات الذكاء الاصطناعي، والذي يُركّز على تعليم أجهزة الحاسب طريقة تفسير العَالَم المرئي وفهمه، ويتضمن استخدام الصور الرقمية ومقاطع الفيديو؛ لتدريب الآلات على التعرّف على المعلومات المرئية وتحليلها مثل: الأشياء والأشخاص والمَشاهد. ويتمثّل الهدف النهائي الذي تسعى رؤية الحاسب إلى تحقيقه في تمكين الآلات من "رؤية" العَالَم كما يراه البشر، واستخدام هذه المعلومات؛ لاتخاذ قرارات، أو للقيام بإجراءات.

هناك مجموعة كبيرة من التطبيقات التي تُستخدم فيها رؤية الحاسب، مثل:

- التصوير الطبي: يمكن أن تساعد رؤية الحاسب الأطباء والمختصين في الرعاية الصحية على تشخيص الأمراض من خلال تحليل الصور الطبية مثل: الأشعّة السينية، والتصوير بالرنين المغناطيسي، والأشعّة المقطعية.
- المركبات ذاتية القيادة: تستخدِم السيارات ذاتية القيادة والطائرات المُسيَّرة رؤية الحاسب للتعرف على إشارات المرور وأشكال الطرق العامة وطرق المشاة والعقبات في الطريق والجو، ولِتمكينها من التنقل بأمان وكفاءة.
- ضبط الجودة: تُستخدم رؤية الحاسب لفحص المنتجات وتحديد عيوب التصنيع، وذلك في مُختلف أنواع الصّناعات، مثل: صناعة السيارات والإلكترونيات والمنسوجات.
- الروبوتية: تُستخدم رؤية الحاسب لمساعدة الروبوتات على التنقل والتفاعل مع بيئتها عن طريق التعرّف على الأشياء والتعامل معها.

يُعدُّ التعلُّم الموجَّه وغير الموجَّه نوعين رئيسين من تعلُّم الآلة يُستخدمان بطريقة شائعة في تطبيقات رؤية الحاسب، ويتضمن كلا النوعين خوارزميات تدريب على مجموعات كبيرة من الصور أو مقاطع الفيديو؛ لكي تتمكن الآلات من التعرّف على المعلومات المرئية وتفسيرها. سبق أن تعرّفت على التعلُّم الموجَّه وغير الموجَّه في الدرسين الأول والثاني من الوحدة الثالثة، وكلاهما طُبِّق في **معالجة اللغات الطبيعية** (NLP) و**توليد اللغات الطبيعية** (NLG)، وسيتم تطبيقهما في هذا الدرس على تحليل الصور.

يتضمَّن التعلُّم غير الموجَّه خوارزميات تدريب على مجموعات بيانات غير مُعَنْوَنة – أي لا توجد فيها عناوين أو فئات صريحة –، ثم تتعلّم الخوارزمية تحديد الأنماط المتشابهة في البيانات دون أن تكون لديها أي معرفة مسبقة بالعناوين. على سبيل المثال: يمكن استخدام خوارزمية التعلُّم غير الموجَّه لتجميع الصور المتشابهة معًا بناءً على السمات المشتركة بينها مثل: اللون أو **النقش** (Texture) أو الشكل. وسيتم توضيح التعلُّم غير الموجَّه بالتفصيل في الدرس الثاني.




شكل 4.1: تصنيف الصور باستخدام رؤية الحاسب

في المقابل، يتضمن التعلُّم الموجَّه تدريب الخوارزميات على مجموعات بيانات مُعَنوَنة؛ حيث يُخصص عنوان أو فئة معيّنة لكل صورة أو مقطع فيديو، ثم تقوم الخوارزمية بعد ذلك بالتعرّف على أنماط وخصائص كل عنوان؛ لتتمكن من تصنيف الصور أو مقاطع الفيديو الجديدة بدقة. فعلى سبيل المثال: قد تُدرَّب خوارزمية التعلُّم الموجَّه على التعرّف على سلالات مُختلفة من القطط بناءً على الصور المُعَنوَنة لكل سلالة (انظر الشكل 4.1)، وسيتم التركيز في هذا الدرس على التعلُّم الموجَّه.

تشتمل عملية التعلُّم الموجَّه عادة على أربع خطوات رئيسة وهي: جمع البيانات، وعَنوَنتها، والتدريب عليها، ثم الاختبار. أثناء جمع البيانات ووضع المسميات، تُجمع الصور أو مقاطع الفيديو وتنظّم في مجموعة بيانات، ثم تُعنون كل صورة أو مقطع فيديو بعنوان صنف أو فئة، مثل: eagle (النسر) أو cat (القطّة).

وتستخدِم خوارزمية تعلُّم الآلة أثناء مرحلة التدريب مجموعة البيانات المُعَنوَنة "لتتعلّم" الأنماط والسمات المرتبطة بكل صنف أو فئة، وكلما زادت بيانات التدريب التي تُقدم للخوارزمية أصبحت أكثر دقة في التعرّف على الفئات المُختلفة في مجموعة البيانات، وبالتالي يتحسَّن أداؤها.

وبمجرد أن يُدرَّب النموذج، يتم اختباره على مجموعة منفصلة غير التي تم التدريب عليها من الصور أو مقاطع الفيديو؛ لتقييم أدائه، وتختلف مجموعة الاختبار عن مجموعة التدريب؛ للتأكد من قدرة النموذج على التعميم على البيانات الجديدة. فعلى سبيل المثال: تحتوي البيانات الخاصة بـ cat (القطّة) على خصائص مثل: الوزن واللون والسلالة وما إلى ذلك، وتُقيّم دقة النموذج بناءً على مدى كفاءة أدائه في مجموعة الاختبار.

تشبه العملية السابقة إلى حد كبير العملية المُتّبعة في مهام التعلُّم الموجَّه لأنواع مُختلفة من البيانات مثل النصوص، ولكن البيانات المرئية عادة ما تُعدُّ أكثر صعوبة في التعامل معها من النصّ لأسباب متعددة كما هو موضَّح في الجدول 4.1.

**جدول 4.1: تحديّات تصنيف البيانات المرئية**

| | |
|---|---|
| البيانات المرئية عالية الأبعاد | تحتوي الصور على كمية كبيرة من البيانات، مما يجعل معالجتها وتحليلها أكثر صعوبة من البيانات النصيَّة، ففي حين أن العناصر الأساسية للمستند النصيّ هي الكلمات، فإن عناصر الصورة هي وحدات البكسل، وسترى في هذا الفصل أنّ الصورة يمكن أن تتكون من آلاف وحدات البكسل، حتّى الصّغيرة منها. |
| البيانات المرئية تحتوي على تفاصيل كثيرة ومتنوّعة للغاية | يمكن أن تتأثر الصور بالتفاصيل الكثيرة، والإضاءة، والتشويش، وعوامل أخرى تجعل تصنيفها بدقة عملية صعبة. بالإضافة إلى ذلك، هناك مجموعة واسعة من البيانات المرئية المتنوّعة ذات العديد من العناصر، والمشاهد، والسياقات التي يصعب تصنيفها بدقة. |
| البيانات المرئية لا تتبع هيكلة محددة | يتبع النصّ بُنية لغوية وقواعد نحويّة عامة، بينما لا تخضع البيانات المرئية لقواعد ثابتة؛ مما يجعل عملية التحليل أكثر تعقيدًا وصعوبة وتكلفة. |

نتيجة لهذه التعقيدات يتطلب التصنيف الفعّال للبيانات المرئية أساليب متخصّصة، وتتناول هذه الوحدة التقنيات التي تستخدِم الخصائص الهندسية واللونية للصور، بالإضافة إلى أساليب تعلُّم الآلة المُتقدمة القائمة على الشبكات العصبية.

يوضِّح الدرس الأول كيفية استخدام لغة **البايثون** (Python) في:

- تحميل مجموعة بيانات من الصور المُعَنوَنة.
- تحويل الصور إلى صيغة رقمية يمكن أن تستخدِمها خوارزميات رؤية الحاسب.
- تقسيم البيانات الرقمية إلى مجموعات بيانات للتدريب، ومجموعات بيانات للاختبار.

• تحليل البيانات؛ لاستخراج أنماط وخصائص مفيدة.
• استخدام البيانات المستخلصة؛ لتدريب نماذج التصنيف التي يمكن استخدامها للتنبؤ بعناوين الصور الجديدة.

تحتوي مجموعة البيانات التي ستستخدمها على ألف وسبعمئة وثلاثين (1,730) صورة لوجوه ستّة عشر نوعًا مُختلفًا من الحيوانات، وبالتالي فهي مجموعة مثالية للتعلُّم الموجَّه لتطبيق التقنيات المذكورة سابقًا.

## تحميل الصور ومعالجتها الأولية Loading and Preprocessing Images

يستورد المقطع البرمجي التالي مجموعة من المكتبات التي تُستخدم لتحميل الصور من مجموعة بيانات LHI-Animal-Faces (وجوه_الحيوانات) وتحويلها إلى صيغة رقمية:

```
%%capture
import matplotlib.pyplot as plt # used for visualization
from os import listdir # used to list the contents of a directory

!pip install scikit-image # used for image manipulation
from skimage.io import imread # used to read a raw image file (e.g. png or jpg)
from skimage.transform import resize # used to resize images

# used to convert an image to the "unsigned byte" format
from skimage import img_as_ubyte
```

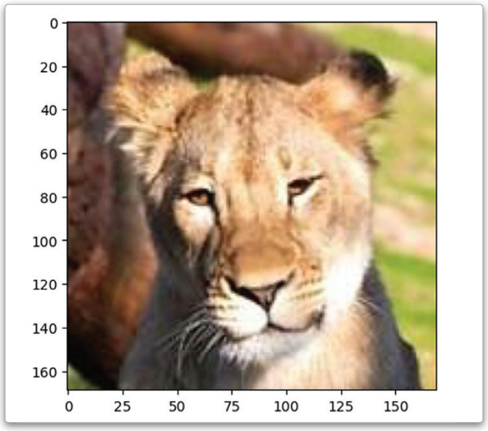
تتطلب خوارزميات التعلُّم الموجَّه أن تكون كل الصور في مجموعة البيانات لها الأبعاد نفسها، ولذلك فإن المقطع البرمجي التالي يقرأ الصور من input_folder (مجلد_المُدخَلات) ويُغيِّر حجم كل منها بحيث تكون لها أبعاد الطول والعرض نفسها:

```
def resize_images(input_folder:str,
                  width:int,
                  height:int
                 ):

    labels = [] # a list with the label for each image
    resized_images = [] # a list of resized images in np array format
    filenames = [] # a list of the original image file names

    for subfolder in listdir(input_folder): # for each sub folder

        print(subfolder)
        path = input_folder + '/' + subfolder

        for file in listdir(path): # for each image file in this subfolder

            image = imread(path + '/' + file) # reads the image
            resized = img_as_ubyte(resize(image, (width, height))) # resizes the image
            labels.append(subfolder[:-4])   # uses subfolder name without "Head" suffix
            resized_images.append(resized) # stores the resized image
            filenames.append(file)          # stores the filename of this image

    return resized_images, labels, filenames
```

```
resized_images, labels, filenames = resize_images("AnimalFace/Image",
width=100, height=100) # retrieves the images with their labels and resizes them to 100 x 100
```

```
BearHead       EagleHead       PigeonHead
CatHead        ElephantHead    RabbitHead
ChickenHead    LionHead        SheepHead
CowHead        MonkeyHead      TigerHead
DeerHead       Natural         WolfHead
DuckHead       PandaHead
```

هذه هي أسماء المجلدات، وبدون المقطع اللاحق Head (رأس)، تُمثِّل هذه الأسماء عناوين للصور الموجودة داخلها.



شكل 4.2: صورة رأس أسد أصلية

تُنشئ دالة imread() تنسيق ألوان للصورة يُعرف بـ "RGB"، ويُستخدم هذا التنسيق على نطاق واسع؛ لأنه يسمح بتمثيل مجموعة واسعة من الألوان. وفي نظام الألوان RGB، تعني الأحرف R و G و B احتواء التنسيق على ثلاثة مكونات رئيسة للألوان، وهي اللون **الأحمر** (R = Red) واللون **الأخضر** (G = Green) واللون **الأزرق** (B = Blue). يُمثِّل كل بكسل بثلاث قنوات وهي: (قناة للون الأحمر، وقناة للون الأخضر، وقناة للون الأزرق)، كل قناة تحوي **ثمانية بت** (8-bit)، ويمكن أن يأخذ البكسل قيمة بين: 0 و255. يُعرف التنسيق 0-255 أيضًا باسم تنسيق **البايت بدون إشارة** (Unsigned Byte).

يتيح الجمع بين هذه القنوات الثلاث تمثيل مجموعة واسعة من الألوان في البكسل، على سبيل المثال: البكسل ذو القيمة (255، 0، 0) سيكون لونه أحمر بالكامل، والبكسل ذو القيمة (0، 255، 0) سيكون لونه أخضر بالكامل، والبكسل ذو القيمة (0، 0، 255) سيكون لونه أزرق بالكامل، والبكسل ذو القيمة (255، 255، 255) سيكون لونه أبيض، والبكسل ذو القيمة (0، 0، 0) سيكون لونه أسود.

في نظام الألوان RGB، تُرتب قيم البكسل في شبكة ثنائية الأبعاد، تحتوي على صفوف وأعمدة تُمثِّل إحداثيات x وy للبكسلات في الصورة، ويُشار إلى هذه الشبكة باسم **مصفوفة الصور** (Image Matrix). على سبيل المثال، ضع في اعتبارك الصورة الموجودة في الشكل 4.2 والمقطع البرمجي المرتبط بها أدناه:

```
# reads an image file, stores it in a variabe and
# shows it to the user in a window
image = imread('AnimalFace/Image/LionHead/lioni78.jpg')
plt.imshow(image)
image.shape
```

```
(169, 169, 3)
```

تكشف طباعة شكل الصورة عن مصفوفة 169 × 169، بإجمالي: ثمانية وعشرين ألفًا وخمسمئة وواحد وستين (28,561) بكسل، ويمثِّل الرقم 3 في العمود الثالث القنوات الثلاث (أحمر / أخضر / أزرق) لنظام الألوان RGB.

على سبيل المثال، سيطبع المقطع البرمجي التالي قيمة الألوان للبكسل الأول من هذه الصورة:

```
# the pixel at the first column of the first row
print(image[0][0])
```

```
[102  68  66]
```

وزارة التعليم
Ministry of Education
2025 - 1447

يؤدي تغيير الحجم إلى تحويل الصور من تنسيق RGB إلى تنسيق **مُستنِد على عدد حقيقي** (Float-Based):

```
resized = resize(image, (100, 100))
print(resized.shape)
print(resized[0][0])
```

```
(100, 100, 3)
[0.40857161 0.27523827 0.26739514]
```

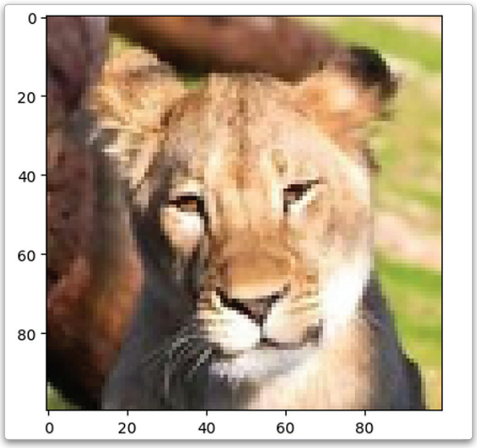
على الرغم من أن الصورة قد غُيّر حجمها إلى مصفوفة ذات أبعاد 100 × 100، فإن قيم القنوات الثلاث RGB لكل بكسل تم **تسويتها** (Normalized) لتكون ذات قيمة بين 0 و1، ويمكن إعادة تحويلها مرة أخرى إلى تنسيق البايت بدون إشارة من خلال المقطع البرمجي التالي:

```
resized = img_as_ubyte(resized)
print(resized.shape)
print(resized[0][0])
print(image[0][0])
```

```
(100, 100, 3)
[104  70  68]
[102  68  66]
```

تختلف قيم الألوان RGB للبكسل الذي غُيّر حجمه اختلافًا بسيطًا عن القيم الموجودة في الصورة الأصلية، وهو من الآثار الشائعة الناتجة عن تغيير الحجم، وعند طباعة الصورة التي غُيّر حجمها، يتبين أنها أقل وضوحًا، كما يظهر في الشكل 4.3، وهذا ناتج عن ضغط المصفوفة 169 × 169 إلى تنسيق 100 × 100.

```
# displays the resized image
plt.imshow(resized);
```



شكل 4.3: صورة رأس أسد غُيّر حجمها

قبل بدء التدريب على خوارزميات التعلُّم الموجَّه، من الجيد التحقق مما إذا كانت أي صورة من الصور الموجودة في مجموعة البيانات غير مطابقة للتنسيق (3، 100، 100).

```
violations = [index for index in range(len(resized_images)) if
resized_images[index].shape != (100,100,3)]

violations
```

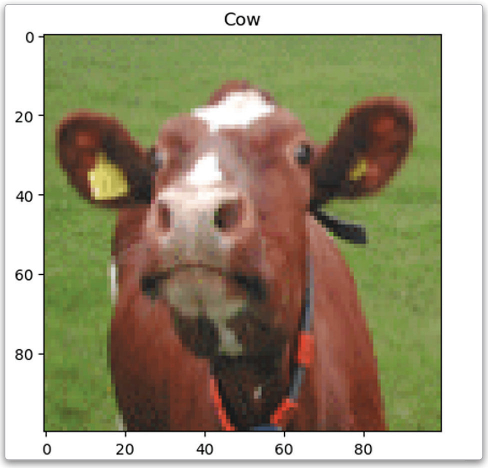
```
[455, 1587]
```

يكشف هذا المقطع البرمجي عن وجود صورتين غير مطابقتين لتلك الصيغة، وهذا غير متوقع؛ لأن دالة ()resize_image تمَّ تطبيقها على جميع الصور الموجودة في مجموعة البيانات. يقوم المقطعان البرمجيان التاليان بطباعة هاتين الصورتين، بالإضافة إلى أبعادهما واسمي ملفيهما:


وزارة التعليم
Ministry of Education
2025 - 1447

```
pos1 = violations[0]
pos2 = violations[1]

print(filenames[pos1])
print(resized_images[pos1].shape)
plt.imshow(resized_images[pos1])
plt.title(labels[pos1])
```

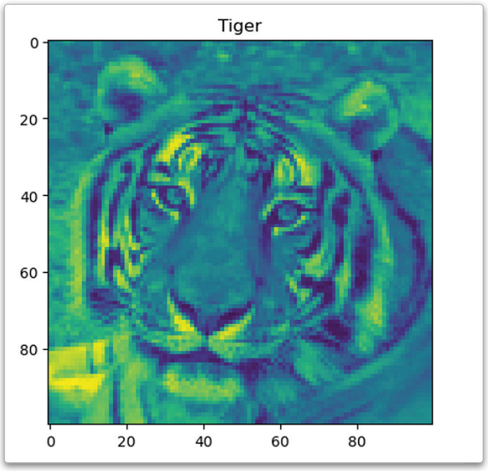
```
cow1.gif
(100, 100, 4)
```




شكل 4.4: صورة بالأحمر والأخضر والأزرق وألفا (RGBA)

```
print(filenames[pos2]);
print(resized_images[pos2].shape);
plt.imshow(resized_images[pos2]);
plt.title(labels[pos2]);
```

```
tiger0000000168.jpg
(100, 100)
```



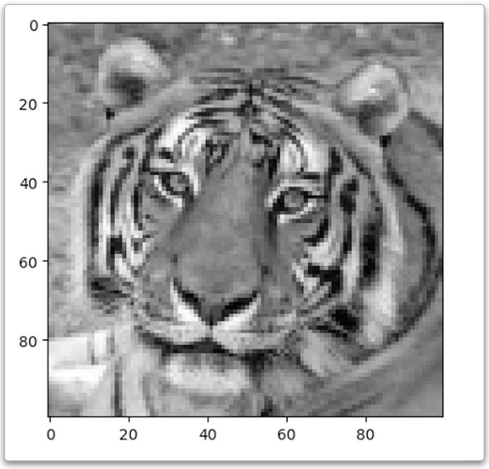

شكل 4.5: صورة تبيّن التنسيق المضلِّل أصفر / أزرق

الصورة الأولى: لها شكل ذو أبعاد (4، 100، 100)، ويدلُّ الرقم 4 أنها بتنسيق RGBA بدلًا من تنسيق RGB، وهذا التنسيق يحتوي على قناة إضافية رابعة تدعى قناة **ألفا** (Alpha) التي تُمثِّل شفافية كل بكسل. على سبيل المثال:

```
# prints the first pixel of the RGBA image
# a value of 255 reveals that the pixel is  not transparent
at all.
resized_images[pos1][0][0]
```

```
array([135, 150, 84, 255], dtype=uint8)
```

الصورة الثانية: لها شكل ذو أبعاد (100، 100)، ويدلُّ غياب البُعد الثالث على أن الصورة بتنسيق **تدرج رمادي** (Grayscale) وليست بتنسيق RGB، والتنسيق **المضلِّل أصفر / أزرق** (Misleading Yellow/Blue) المبين سابقًا يعود إلى خريطة لونية تُطبِّقها الدالة imshow بشكل افتراضي على الصور ذات التدرج الرمادي، ويمكن إلغاؤه كما يلي:




شكل 4.6: صورة بتدرج رمادي

```
plt.imshow(resized_images[pos2], cmap = 'gray')
```

صور التدرج الرمادي لها قناة واحدة فقط (بدلًا من قنوات RGB الثلاث)، وقيمة كل بكسل عبارة عن رقم واحد يتراوح من 0 إلى 255، حيث تُمثِّل قيمة البكسل 0 اللون الأسود، بينما تُمثِّل قيمة البكسل 255 اللون الأبيض. على سبيل المثال:

```
resized_images[pos2][0][0]
```

```
100
```

وكاختبار إضافي لجودة البيانات، يقوم المقطع البرمجي التالي بحساب تكرار عنوان كل صورة حيوان في مجموعة البيانات:

```
# used to count the frequency of each element in a list.
from collections import Counter

label_cnt = Counter(labels)
label_cnt
```

```
Counter({'Bear': 101,
         'Cat': 160,
         'Chicken': 100,
         'Cow': 104,
         'Deer': 103,
         'Duck': 103,
         'Eagle': 101,
         'Elephant': 100,
         'Lion': 102,
         'Monkey': 100,
         'Nat': 8,
         'Panda': 119,
         'Pigeon': 115,
         'Rabbit': 100,
         'Sheep': 100,
         'Tiger': 114,
         'Wolf': 100})
```

هنا يمكنك رؤية القيمة المتطرفة وهي فئة Nat (Nature أو الطبيعة)، وتحتوي على ثمانية عناصر فقط مقارنة بالفئات الأخرى.

تحتوي مجموعة البيانات على صور حيوانات وصور أخرى من الطبيعة؛ وذلك بهدف التعرّف على الصور التي تشذ عن صور الحيوانات. يكشف Counter (العداد) عن فئة صغيرة جدًا عنوانها Nat (الطبيعة)، وتحتوي على ثماني صور فقط، وعندما تقوم بكشف سريع يتضح لك أن هذه الفئة ذات **قيم متطرفة** (Outlier) تحتوي على صور لمناظر طبيعية ولا يوجد بها أي وجه لأي حيوان.

يقوم المقطع البرمجي التالي بإزالة صورة RGBA وصورة التدرج الرمادي، وكذلك كل الصور التي تنتمي لفئة Nat (الطبيعة) من قوائم أسماء الملفات، والعناوين، والصور التي غُيِّر حجمها.

```
N = len(labels)

resized_images = [resized_images[i] for i in range(N) if i not in violations
and labels[i] != "Nat"]
filenames = [filenames[i] for i in range(N) if i not in violations and
labels[i] != "Nat"]
labels = [labels[i] for i in range(N) if i not in violations and labels[i] !=
"Nat"]
```

تتمثّل الخطوة التالية في تحويل resized_images (الصور _ المُعدَّل حجمها) وقوائم العناوين إلى مصفوفات Numpy (نمباي) حسب ما تتوقعه العديد من خوارزميات رؤية الحاسب. يستخدِم المقطع البرمجي التالي أيضًا المتغيِّرات (X، Y) التي تُستخدم في العادة لتمثيل البيانات والعناوين على التوالي في مهام التعلُّم الموجَّه:

```
import numpy as np
X = np.array(resized_images)
Y = np.array(labels)

X.shape
```

```
(1720, 100, 100, 3)
```

يوضِّح شكل مجموعة بيانات X النهائية اشتمالها على ألف وسبعمئة وعشرين صورة بتنسيق RGB، بناءً على عدد القنوات، وجميعها بأبعاد 100 × 100 (أي عشرة آلاف بكسل). أخيرًا، يمكن استخدام دالة ()train_test_split من مكتبة sklearn لتقسيم مجموعة البيانات إلى مجموعة تدريب ومجموعة اختبار.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size = 0.20,    # uses 20% of the data for testing
    shuffle = True,      # to randomly shuffle the data.
    random_state = 42,   # to ensure that data is always shuffled in the same way
)
```

نظرًا لأن مجلدات صور الحيوانات حُمِّلت مجلّدًا تلو الآخر، فإن الصور من كل مجلد جُمعت معًا في القوائم السابقة، وقد يؤدي ذلك إلى تضليل العديد من الخوارزميات، خاصة في مجال رؤية الحاسب، وضبط shuffle=True (تفعيل إعادة الترتيب) في المقطع البرمجي السابق يحل هذه المشكلة، وبوجه عام، من الجيد إعادة ترتيب البيانات عشوائيًا قبل إجراء أي تحليل.

## التنبؤ بدون هندسة الخصائص Prediction without Feature Engineering

على الرغم من أن الخطوات المتَّبعة في القسم السابق قد حوَّلت البيانات إلى تنسيق رقمي، إلا أنه ليس بالتنسيق القياسي أحادي البُعد الذي تتوقعه العديد من خوارزميات تعلُّم الآلة. على سبيل المثال، وصفتْ الوحدة الثالثة كيف يجب تحويل كل مستند إلى متَّجه رقمي أحادي البُعد قبل استخدام البيانات في تدريب نماذج تعلُّم الآلة واختبارها، بينما تحتوي كل نقطة بيانات في مجموعة البيانات المرئية هنا على تنسيق ثلاثي الأبعاد.

```
X_train[0].shape
```

```
(100, 100, 3)
```

لذلك يمكن استخدام المقطع البرمجي التالي **لتسطيح** (Flatten) كل صورة في متَّجَه أحادي البُعد، فكل صورة الآن ممثَّلة كمتَّجَه رقمي مسطح قيمته 3 × 100 × 100 = 30,000 قيمة.

```
X_train_flat = np.array([img.flatten() for img in X_train])
X_test_flat = np.array([img.flatten() for img in X_test])
X_train_flat[0].shape
```

```
(30000,)
```

يمكن استخدام هذا التنسيق المسطح مع أي خوارزمية تصنيف قياسية دون بذل أي جهد إضافي لهندسة خصائص تنبؤية أخرى، وسيوضِّح القسم التالي مثالًا على هندسة الخصائص لبيانات صورة، ويستخدِم المقطع البرمجي التالي مُصنِّف **بايز الساذج** (Naive Bayes – NB) الذي استُخدم أيضًا لتصنيف البيانات النصيَّة في الوحدة الثالثة:

```
from sklearn.naive_bayes import MultinomialNB # imports the Naive Bayes Classifier

model_MNB = MultinomialNB()
model_MNB.fit(X_train_flat,y_train) # fits the model on the flat training data
```

```
MultinomialNB()
```

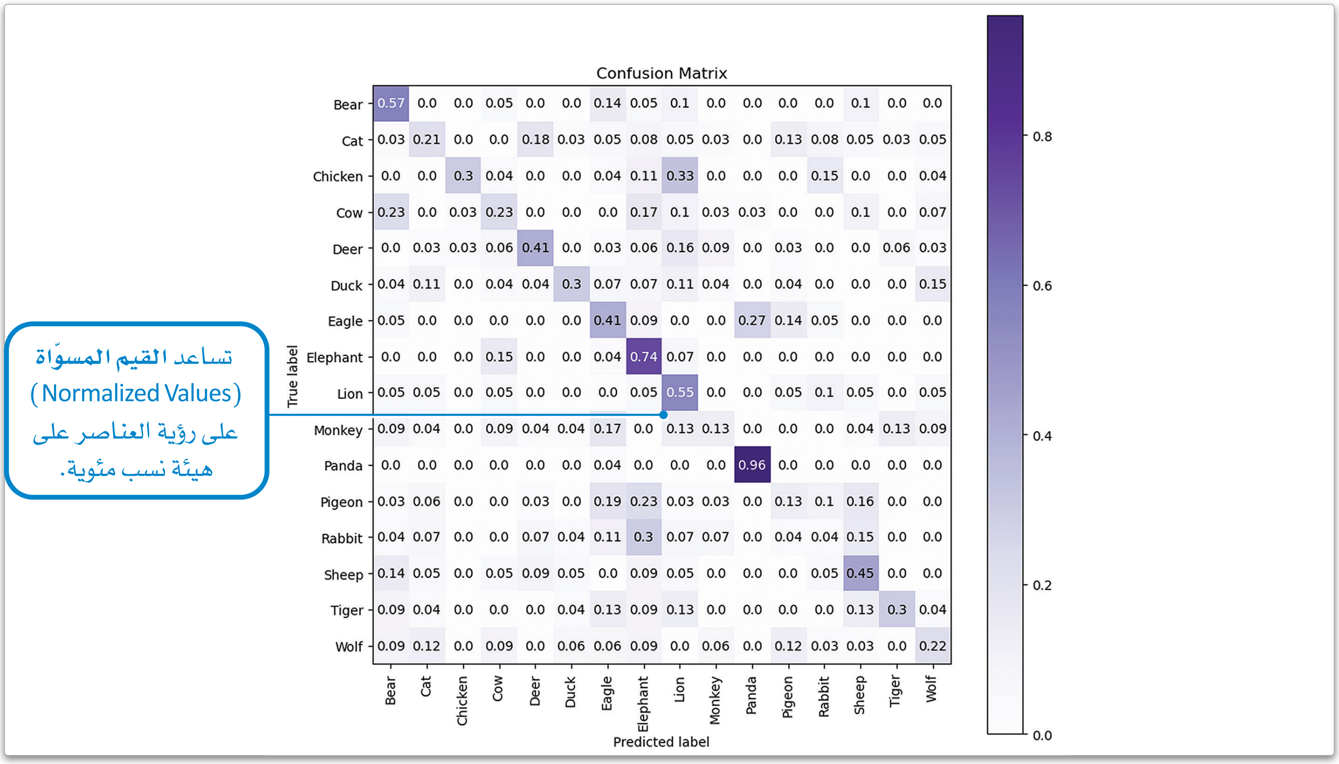
```
from sklearn.metrics import accuracy_score # used to measure the accuracy

pred = model_MNB.predict(X_test_flat) # gets the predictions for the flat test set
accuracy_score(y_test,pred)
```

```
0.36046511627906974
```

يعرض المقطع البرمجي التالي **مصفوفة الدقة** (Confusion Matrix) الخاصة بالنتائج لإعطاء رؤية إضافية:

```
%%capture
!pip install scikit-plot
import scikitplot
```

```
scikitplot.metrics.plot_confusion_matrix(y_test, # actual labels
                                    pred, # predicted labels
                                    title = "Confusion Matrix",
                                    cmap = "Purples",
                                    figsize = (10,10),
                                    x_tick_rotation = 90,
                                    normalize = True # to print percentages
                                    )
```




شكل 4.7: مصفوفة الدقة الخاصة بأداء خوارزمية MultinomialNB

تُحقق خوارزمية **بايز الساذجة متعددة الحدود** (MultinomialNB) دقة تقارب 30%، وعلى الرغم من أن هذه النسبة قد تبدو قليلة، إلا أن عليك النظر إليها في ضوء أن مجموعة البيانات تتضمن عشرين عنوانًا مُختلفًا. ويعني ذلك أنّه لو افتُرض وجود مجموعة بيانات متوازنة نسبيًا يُغطى فيها كل عنوان 20/1 من البيانات، فإن المُصنِّف العشوائي الذي يُخصص عنوانًا لكل نقطة اختبار بشكل عشوائي، سيحقق دقة تبلغ حوالي 5%، ولذلك ستكون الدقة بنسبة 30% أعلى بست مرات من التخمين العشوائي.

ومع ذلك، كما هو موضَّح في الأقسام التالية، يمكن تحسين هذه الدقة تحسينًا ملحوظًا، وتؤكد مصفوفة الدقة أيضًا أن هناك مجالًا للتحسين. على سبيل المثال، غالبًا ما يخطئ نموذج بايز الساذج ويصنِّف Pigeons (الحَمَام) على أنها Eagles (نسور) أو يصنِّف Wolves (الذئاب) على أنها Cats (قطط). تكمن أسهل طريقة لمحاولة تحسين النتائج في ترك البيانات كما هي، والتجريب باستخدام مُصنِّفات مُختلفة، ومن النماذج التي ثبت أنها تعمل بشكل جيد مع بيانات الصورة المحوَّلة إلى متَّجَهَات نموذج: **مُصنِّف الانحدار التَّدرجي العشوائي** (SGDClassifier) من مكتبة Sklearn، حيث يعمل نموذج SGDClassifier أثناء التدريب على ضبط أوزان النموذج بناءً على بيانات التدريب، والهدف من ذلك يتمثّل في العثور على مجموعة الأوزان التي تقلل من **دالة الخسارة** (Loss Function)، وهي الدالة التي تقيس الفرق بين العناوين المتوقَّعة والعناوين الحقيقية في بيانات التدريب.

**خوارزمية بايز الساذجة متعددة الحدود (MultinomialNB Algorithm):**

هي خوارزمية تعلُّم آلة تُستخدم لتصنيف النصوص أو البيانات الأخرى في فئات مُختلفة، وتعتمد على خوارزمية **بايز الساذج** (Naive Bayes) وهي طريقة بسيطة وفعّالة لحل مشكلات التصنيف.

**خوارزمية مُصنِّف الانحدار التَّدرجي العشوائي (SGDClassifier Algorithm):**

هي خوارزمية تعلُّم آلة تُستخدم في تصنيف البيانات في فئات مُختلفة أو مجموعات، وتعتمد على أسلوب يسمى **الانحدار التَّدرجي العشوائي** (Stochastic Gradient Descent – SGD)، وهي طريقة فعّالة لتحسين الأنواع المتعددة للنماذج وتدريبها، بما فيها المُصنِّفات.

يستخدِم المقطع البرمجي التالي مُصنِّف SGDClassifier لتدريب نموذج على مجموعة بيانات مسطحة:



وزارة التعليم
Ministry of Education
2025 - 1447

```
from sklearn.linear_model import SGDClassifier

model_sgd = SGDClassifier()
model_sgd.fit(X_train_flat, y_train)
pred=model_sgd.predict(X_test_flat)
accuracy_score(y_test,pred)
```

```
0.46511627906976744
```

**التحجيم القياسي (Standard Scaling):**
هو تقنية معالجة أولية تُستخدم في تعلُّم الآلة لتحجيم خصائص مجموعة البيانات بحيث تكون ذات متوسط حسابي صفري وتباين أحادي الوحدة.

يُحقق مصنِّف SGDClassifier دقة أعلى بشكل ملحوظ تزيد عن 46%، على الرغم من تدريبه على البيانات نفسها التي دُرِّب مُصنِّف MultinomialNB عليها، ويدل ذلك على فائدة تجربة خوارزميات تصنيف مُختلفة؛ للعثور على أفضل خوارزمية تتناسب مع أي مجموعة بيانات مُعطاة، ومن المهم فهم نقاط القوة والضعف لكل خوارزمية، فعلى سبيل المثال: من المعروف أن خوارزمية SGDClassifier تعمل بشكل أفضل عندما تُحجّم بيانات الإدخال وتُوحّد الخصائص؛ ولهذا السبب ستستخدِم التحجيم القياسي في نموذجك.

يستخدِم المقطع البرمجي التالي أداة StandardScaler (المُحجِّم القياسي) من مكتبة sklearn لتحجيم البيانات:

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_flat_scaled = scaler.fit_transform(X_train_flat)
X_test_flat_scaled = scaler.fit_transform(X_test_flat)

print(X_train_flat[0]) # the values of the first image pre-scaling
print(X_train_flat_scaled[0]) # the values of the first image post-scaling
```

```
[144 142 151 ...  76  75  80]
[ 0.33463473  0.27468959  0.61190285 ... -0.65170221 -0.62004162
 -0.26774175]
```

يمكن الآن تدريب نموذج جديد واختباره باستخدام مجموعات البيانات التي تم تحجيمها:

```
model_sgd = SGDClassifier()
model_sgd.fit(X_train_flat_scaled, y_train)
pred=model_sgd.predict(X_test_flat_scaled)
accuracy_score(y_test,pred)
```

```
0.4906976744186046
```

تدل النتائج على وجود تحسُّن بعد التحجيم، ومن المحتمل أن يحدث تحسين إضافي بواسطة تجريب خوارزميات أخرى وضبط متغيِّراتها حتّى تتناسب مع مجموعة البيانات بشكل أفضل.

## التنبؤ بانتقاء الخصائص Prediction with Feature Selection

ركَّز القسم السابق على تدريب النماذج عن طريق تسطيح البيانات، في حين سيصف هذا القسم كيفية تحويل البيانات الأصلية لهندسة الخصائص الذكية التي تلتقط الصفات الرئيسة لبيانات الصورة، وعلى وجه التحديد يوضِّح القسم تقنية شائعة تسمى **المُخطَّط التكراري للتدرجات الموجَّهة** (Histogram of Oriented Gradients –HOG).

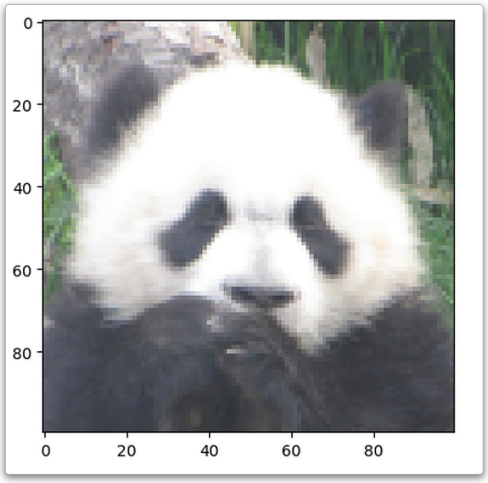
**المُخطَّطات التكرارية للتدرجات الموجَّهة (Histogram of Oriented Gradients -HOG):**

تقوم المُخطَّطات التكرارية للتدرجات الموجَّهة بتقسيم الصورة إلى أقسام صغيرة وتحلِّل توزيع تغيرات الكثافة في كل قسم حتّى تحدِّد وتفهم شكل الكائن في الصورة.

تتمثّل الخطوة الأولى في هندسة المُخطَّطات التكرارية للتدرجات الموجَّهة في تحويل الصور من تنسيق RGB إلى صور ذات تدرج رمادي، ويمكن القيام بذلك باستخدام الدالة rgb2gray() من مكتبة sckit-image:

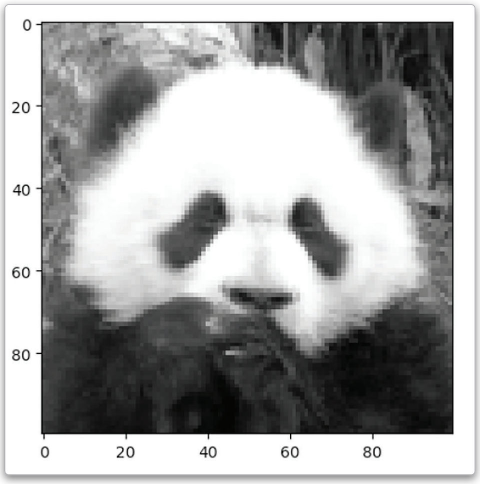
```
from skimage.color import rgb2gray # used to convert a multi-color (rgb) image to grayscale
# converts the training data
X_train_gray = np.array([rgb2gray(img) for img in X_train])
# converts the testing data
X_test_gray = np.array([rgb2gray(img) for img in X_test])
```

```
plt.imshow(X_train[0]);
```



شكل 4.8: صورة بالألوان الأساسية

```
plt.imshow(X_train_gray[0],cmap='gray');
```



شكل 4.9: صورة ذات تدرج رمادي

الشكل الجديد لكل صورة أصبح بتنسيق 100 × 100، بدلًا من التنسيق RGB المُستنِد إلى 100 × 100×3:

```
print(X_train_gray[0].shape)
print(X_train[0].shape)
```

```
(100, 100)
(100, 100, 3)
```



وزارة التعليم
Ministry of Education
2025 - 1447

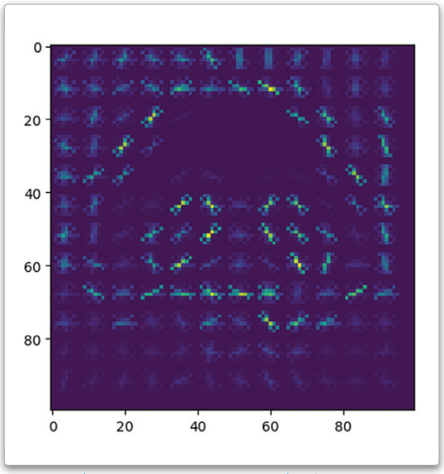
تتمثّل الخطوة التالية في إنشاء خصائص المُخطَّط التكراري للتدرجات الموجَّهة لكل صورة في البيانات، ويمكن تحقيق ذلك من خلال دالة ( )hog من مكتبة scikit-image، ويوضِّح المقطع البرمجي التالي مثالًا على الصورة الأولى في مجموعة بيانات التدريب:

```
from skimage.feature import hog

hog_vector, hog_img = hog(
                    X_train_gray[0],
                    visualize = True
                    )
hog_vector.shape
```

```
(8100,)
```

hog_vector هو متَّجَه أحادي البُعد ذو ثمانية آلاف ومئة قيمة عددية، ويمكن استخدامها لتمثيل الصورة، ويَظهر التمثيل البصري لهذا المتَّجَه باستخدام:

```
plt.imshow(hog_img);
```



شكل 4.10: مُخطَّط تكراري للتدرجات الموجَّهة لصورة

يصوِّر هذا التمثيل الجديد حدود الأشكال الأساسية في الصورة، ويحذف التفاصيل الأخرى ويُركِّز على الأجزاء المفيدة التي يمكنها أن تساعد المُصنِّف على أن يقوم بالتنبؤ، ويطبِّق المقطع البرمجي التالي هذا التغيير على كل الصور في كل من مجموعة التدريب ومجموعة الاختبار:

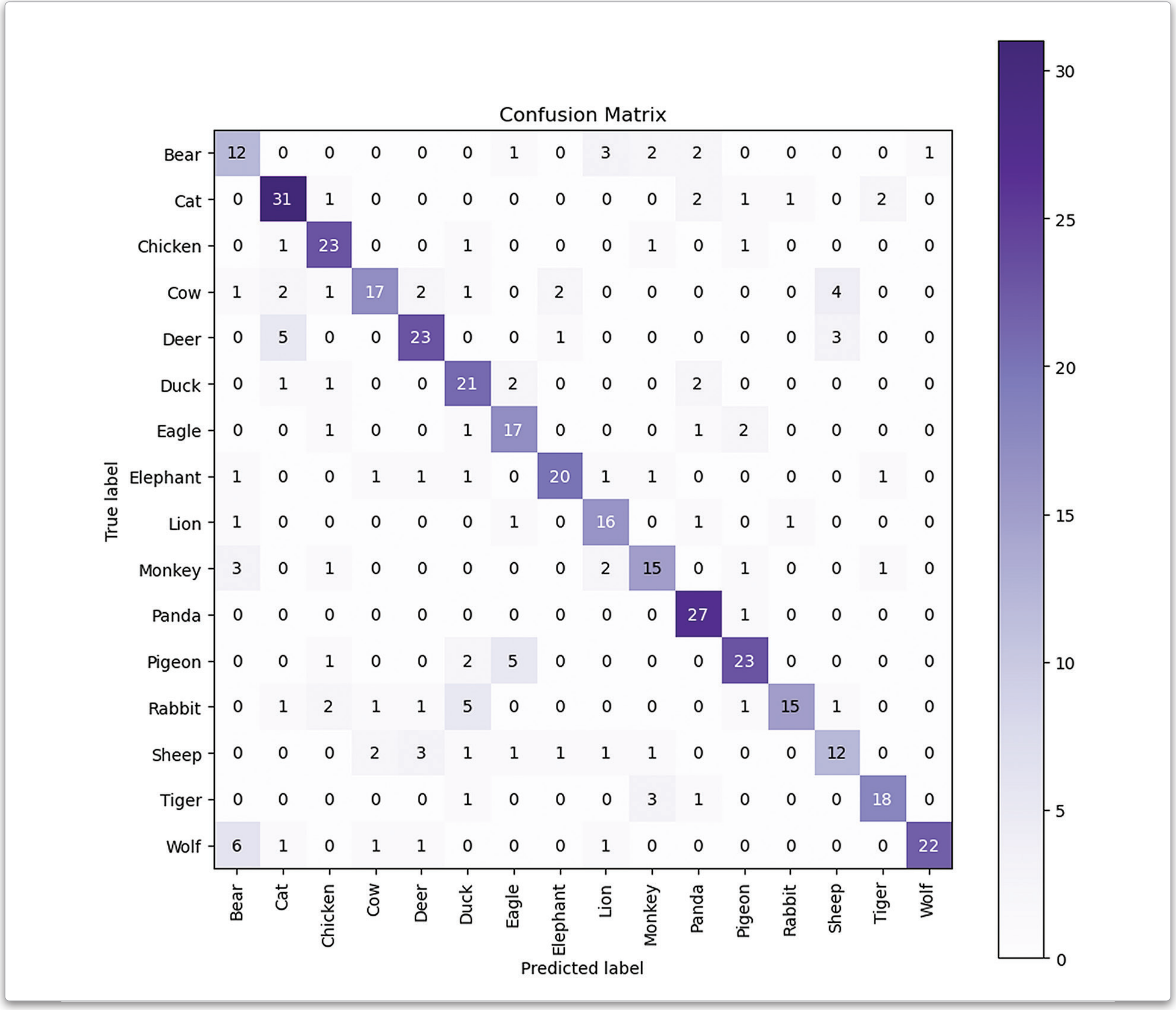
```
X_train_hog = np.array([hog(img) for img in X_train_gray])
X_test_hog = np.array([hog(img) for img in X_test_gray])
```

يمكن الآن تدريب SGDClassifier على هذا التمثيل الجديد:

```
# scales the new data
scaler = StandardScaler()
X_train_hog_scaled = scaler.fit_transform(X_train_hog)
X_test_hog_scaled = scaler.fit_transform(X_test_hog)

# trains a new model
model_sgd = SGDClassifier()
model_sgd.fit(X_train_hog_scaled, y_train)

# tests the model
pred = model_sgd.predict(X_test_hog_scaled)
accuracy_score(y_test,pred)
```

```
0.7418604651162791
```

```python
scikitplot.metrics.plot_confusion_matrix(y_test, # actual labels
                                         pred, # predicted labels
                                         title = "Confusion Matrix", # title to use
                                         cmap = "Purples", # color palette to use
                                         figsize = (10,10), # figure size
                                         x_tick_rotation = 90
                                         );
```




شكل 4.11: مصفوفة الدقة لأداء خوارزمية SGDClassifier

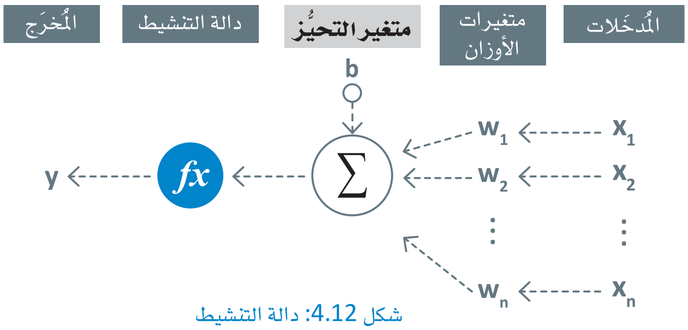
تكشف النتائج الجديدة عن تحسُّن هائل في الدقة التي قفزت لتصل إلى أكثر من 70 %، وتجاوزت بكثير الدقة التي حققها المُصنِّف نفسه على البيانات المسطحة دون القيام بأي هندسة للخصائص، ويتضح التحسُّن أيضًا في مصفوفة الدقة المُحدَّثة التي تشمل عددًا أقل من الأخطاء (التنبؤات الإيجابية الخاطئة)، ويوضِّح ذلك أهمية استخدام تقنيات رؤية الحاسب لهندسة خصائص ذكية تلتقط الصفات المرئية المُختلفة للبيانات.

## التنبؤ باستخدام الشبكات العصبية Prediction Using Neural Networks

يوضِّح هذا القسم كيفية استخدام الشبكات العصبية لتصميم مُصنِّفات مخصصة لبيانات الصور، وكيف يمكنها في كثير من الأحيان أن تتفوّق على التقنيات عالية الفعالية مثل: عملية المُخطَّط التكراري للتدرجات الموجَّهة التي وُصفت في القسم السابق، وتُستخدم مكتبة TensorFlow ومكتبة Keras الشهيرتان لهذا الغرض.

مكتبة TensorFlow هي مكتبة منخفضة المستوى تُوفِّر مجموعة واسعة من أدوات تعلُّم الآلة والذكاء الاصطناعي، وتسمح للمستخدِمين بتعريف الحسابات العددية التي تتضمن **متَّجهات متعددة الأبعاد** (Tensors) ومعالجتها، وهي مصفوفات متعددة الأبعاد من البيانات. من ناحية أخرى، تُعدُّ مكتبة Keras ذات مستوى أعلى وتُوفِّر واجهة أبسط لبناء النماذج وتدريبها، وهي مبنية باستخدام مكتبة TensorFlow (أو مكتبات خلفية أخرى) وتُوفِّر مجموعة من الطبقات والنماذج المعرَّفة مسبقًا والتي يمكن تجميعها بسهولة لبناء نموذج تعلُّم عميق. وصُمِّمت مكتبة Keras لتكون صديقة للمستخدِم وسهلة الاستخدام؛ مما يجعلها خيارًا رائجًا للممارسين.

**دوال التنشيط** (Activation Functions) هي دوال رياضية تُطَبَّق على مُخرَجات كل خلية عصبية في الشبكة العصبية، كما تتميز بأنها تضيف خصائصَ **غير خطية** (Non-linear) للنموذج وتسمح للشبكة بتعلُّم الأنماط المعقدة في البيانات، ويُعدُّ اختيار دالة التنشيط أمرًا مهمًا ويمكن أن يؤثر على أداء الشبكة، حيث تتلقى الخلايا العصبية المُدخَلات وتعالجها من خلال متغيرات الأوزان والتحيُّزات وتنتج مُخرَجات بناء على دالة التنشيط كما يظهر في الشكل 4.12. تُنشأ الشبكات العصبية من خلال ربط العديد من الخلايا العصبية معًا في طبقات، وتُدرَّب على ضبط متغيرات الأوزان والتحيُّزات وتحسين أدائها بمرور الوقت.




شكل 4.12: دالة التنشيط

يُثبِّت المقطع البرمجي التالي مكتبة tensorflow ومكتبة keras:

```
%%capture
!pip install tensorflow
!pip install keras
```

في الوحدة السابقة، تعرّفت على الخلايا العصبية الاصطناعية وعلى معماريات الشبكات العصبية، وعلى وجه التحديد تعرّفت على نموذج **الكلمة إلى المتَّجه** (Word2Vec) الذي يَستخدِم طبقة مخفية وطبقة مُخرَجات؛ ليتنبأ بسياق الكلمات لكلمة مُعطاة في جملة. وبعد ذلك تُستخدم مكتبة Keras لإنشاء معمارية عصبية مشابهة للصور. أولًا: تُحوَّل العناوين في y_train إلى تنسيق أعداد صحيحة، طبقًا لمتطلبات مكتبة Keras.

```
# gets the set of all distinct labels
classes=list(set(y_train))
print(classes)
print()

# replaces each label with an integer (its index in the classes lists) for both the training and testing data
y_train_num = np.array([classes.index(label) for label in y_train])
y_test_num = np.array([classes.index(label) for label in y_test])
print()

#example:
print(y_train[:5]) # first 5 labels
print(y_train_num[:5]) # first 5 labels in integer format
```

```
['Elephant', 'Duck', 'Monkey', 'Cow', 'Sheep', 'Wolf', 'Tiger', 'Deer',
'Cat', 'Lion', 'Rabbit', 'Panda', 'Pigeon', 'Chicken', 'Eagle', 'Bear']

['Panda' 'Pigeon' 'Monkey' 'Panda' 'Sheep']
[11 12 2 11 4]
```

ويمكن الآن استخدام أداة Sequential (التتابع) من مكتبة Keras لبناء شبكة عصبية في شكل طبقات متتابعة.

```
from keras.models import Sequential  # used to build neural networks as sequences of layers
# every neuron in a dense layer is connected to every other neuron in the previous layer.
from keras.layers import Dense

# builds a sequential stack of layers
model = Sequential()
# adds a dense hidden layer with 200 neurons, and the ReLU activation function.
model.add(Dense(200,input_shape = (X_train_hog.shape[1],), activation='relu'))
# adds a dense output layer and the softmax activation function.
model.add(Dense(len(classes), activation='softmax'))
model.summary()
```

```
Model: "sequential"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 200)               1620200

 dense_1 (Dense)             (None, 16)                3216

=================================================================
Total params: 1,623,416
Trainable params: 1,623,416
Non-trainable params: 0
_________________________________________________________________
```

عدد الخلايا العصبية في الطبقة المخفية يعتمد على الخيار الذي يُتخذ عند التصميم، وعدد الفئات يحدِّد عدد الخلايا العصبية في طبقة المُخرَجات.

يكشف ملخص النموذج عن العدد الإجمالي للمتغيِّرات التي يجب أن يتعلّمها النموذج من خلال ضبطها على بيانات التدريب، وبما أن المُدخَلات تحتوي على ثمانية آلاف ومئة (8,100) مُدخَل، وهي أبعاد صور المُخطَّط التكراري للتدرجات الموجَّهة X_train_hog وتحتوي الطبقة المخفية على مئتي خلية عصبية، وهي طبقة كثيفة متصلة بالمُدخَلات اتصالًا كاملًا، فإن المجموع 8,100 × 200 = 1,620,000 وصلة موزونة يجب تعلُّم أوزانها (متغيِّراتها). تمت إضافة مئتي متغيِّر **تحيُّز** (Bias) إضافي، بواقع متغيِّر لكل خلية عصبية في الطبقة المخفية، ومتغيِّر التحيُّز هو قيمة تُضاف إلى مُدخَلات كل خلية عصبية في الشبكة العصبية، وتُستخدم لتوجيه دالة تنشيط الخلايا العصبية إلى الجانب السلبي أو الإيجابي، مما يسمح للشبكة بنمذجة علاقات أكثر تعقيدًا بين بيانات المُدخَلات وعناوين المُخرَجات.



وزارة التعليم
Ministry of Education
2025 - 1447

وبما أن طبقة المُخرَجات تحتوي على ستّ عشرة خلية عصبية متصلة بالكامل بمئتي خلية عصبية موجودة في الطبقة المخفية، فإن مجموع الوصلات الموزونة يبلغ 16 × 200 = 3,200. ويُضاف ستة عشر متغيّر تحيُّز إضافي، بواقع متغيِّر واحد لكل خلية عصبية في طبقة المُخرَجات، ويُستخدم السطر البرمجي التالي **لتجميع** (Compile) النموذج:

```
# compiling the  model
model.compile(loss = 'sparse_categorical_crossentropy', metrics =
['accuracy'], optimizer = 'adam')
```

تُستخدم دالة إعداد النموذج الذكي في مكتبة Keras والمعروفة **بالتجميع** (model.compile()) في عملية تحديد الخصائص الأساسية للنموذج الذكي وإعداده للتدريب والتحقق والتنبؤ، وتتخذ ثلاثة مُعامِلات رئيسة كما هو موضَّح في الجدول 4.2.

**جدول 4.2: مُعامِلات طريقة التجميع**

| | |
|---|---|
| **الخسارة** (loss) | هي الدالة التي تُستخدم لتقييم الخطأ في النموذج أثناء التدريب، وتقيس مدى تطابق تنبؤات النموذج مع العناوين الحقيقية لمجموعة معيّنة من بيانات المُدخَلات. الهدف من التدريب تقليل دالة الخسارة مما يتضمن في العادة تعديل أوزان النموذج ومقدار التحيُّز، وفي هذه الحالة تكون دالة الخسارة هي: sparse_categorical_crossentropy وهي دالة خسارة مناسبة لمهام التصنيف متعدّدة الفئات؛ حيث تكون العناوين أعدادًا صحيحة كما في y_train_num. |
| **المقاييس** (metrics) | هي قائمة المقاييس المستخدَمة لتقييم النموذج أثناء التدريب والاختبار، وتُحسب هذه المقاييس باستخدام مُخرَجات النموذج والعناوين الحقيقية، ويمكن استخدامها لمراقبة أداء النموذج وتحديد المجالات التي يمكن تحسينه فيها. مقياس **الدقة** (Accuracy) هو مقياس شائع لمهام التصنيف يقيس نسبة التنبؤات الصحيحة التي قام بها النموذج. |
| **المُحسِّن** (optimizer) | هو خوارزمية التحسين التي تُستخدم في ضبط أوزان النموذج ومقدار التحيُّز أثناء التدريب. ويستخدِم المحسِّن دالة الخسارة والمقاييس لإرشاد عملية التدريب، ويقوم بضبط متغيِّرات النموذج في محاولة لتقليل الخسارة وزيادة أداء النموذج إلى الحد الأقصى. وفي هذه الحالة فقد تم استخدام المُحسِّن adam الذي يُعدُّ خوارزمية شائعة لتدريب الشبكات العصبية. |

وأخيرًا، تُستخدم دالة fit () لتدريب النموذج على البيانات المتاحة.

```
model.fit(X_train_hog, # training data
          y_train_num, # labels in integer format
          batch_size = 80, # number of samples processed per batch
          epochs = 40, # number of iterations over the whole dataset
         )
```

```
Epoch 1/40
17/17 [==============================] - 1s 16ms/step - loss: 2.2260 - accuracy: 0.3333
Epoch 2/40
17/17 [==============================] - 0s 15ms/step - loss: 1.1182 - accuracy: 0.7256
Epoch 3/40
17/17 [==============================] - 0s 15ms/step - loss: 0.7198 - accuracy: 0.8155
Epoch 4/40
17/17 [==============================] - 0s 15ms/step - loss: 0.4978 - accuracy: 0.9031
Epoch 5/40
17/17 [==============================] - 0s 16ms/step - loss: 0.3676 - accuracy: 0.9388
...
Epoch 36/40
17/17 [==============================] - 0s 15ms/step - loss: 0.0085 - accuracy: 1.0000
Epoch 37/40
17/17 [==============================] - 0s 21ms/step - loss: 0.0080 - accuracy: 1.0000
Epoch 38/40
17/17 [==============================] - 0s 15ms/step - loss: 0.0076 - accuracy: 1.0000
Epoch 39/40
17/17 [==============================] - 0s 15ms/step - loss: 0.0073 - accuracy: 1.0000
Epoch 40/40
17/17 [==============================] - 0s 15ms/step - loss: 0.0071 - accuracy: 1.0000
```

تُستخدم دالة fit ( ) لتدريب نموذج على مجموعة معيّنة من بيانات الإدخال والعناوين، وتتخذ أربع مُعامِلات رئيسة، كما هو موضّح في الجدول 4.3.

**جدول 4.3: مُعامِلات طريقة fit**

| | |
|---|---|
| X_train_hog | هو مُعامِل بيانات الإدخال المستخدَمة لتدريب النموذج، وتتكون من البيانات المحوَّلة عن طريق المُخطَّط التكراري للتدرجات الموجَّهة التي استُخدمت أيضًا لتدريب أحدث إصدار من خوارزمية SGDClassifier في القسم السابق. |
| y_train_num | هو مُعامِل يتضمّن عنوانًا لكل صورة بتنسيق أعداد صحيحة. |
| batch_size | هو عدد العينات التي تمت معالجتها في كل دُفعة أثناء التدريب، ويقوم النموذج بتحديث أوزانه ومقدار التحيُّز بعد كل دُفعة، ويمكن أن يؤثر حجم الدُفعة على سرعة عملية التدريب، واستقرارها، كما يمكن أن تؤدي أحجام الدُفعات الأكبر إلى تدريب أسرع، ولكنها قد تكون أكثر تكلفة من الناحية الحسابية وقد تؤدي إلى تدرجات أقل استقرارًا. |
| epochs | هو عدد المرات التي يتكرر فيها تدريب النموذج باستخدام مجموعة البيانات بأكملها، وتتكون **الفترة** (Epoch) من مرور واحد عبر مجموعة البيانات بأكملها. ويقوم النموذج بتحديث أوزانه ومقدار التحيُّز بعد كل دورة، كما يمكن أن يؤثر عدد الفترات على قدرة النموذج على التعلُّم والتعميم على البيانات الجديدة، والفترة متغيِّر مهم يجب اختياره بعناية، وفي هذه الحالة يُدرَّب النموذج على أربعين فترة. |

وزارة التعليم
Ministry of Education
2025 - 1447

ويمكن الآن استخدام نموذج التدريب للتنبؤ بعناوين الصور في مجموعة الاختبار.

```
pred = model.predict(X_test_hog)
pred[0] # prints the predictions for the first image
```

```
14/14 [==============================] - 0s 2ms/step

array([4.79123509e-03, 9.79321003e-01, 8.39506648e-03, 1.97884417e-03,
       7.83501855e-06, 3.50346789e-04, 3.45465224e-07, 1.19854585e-05,
       4.41945267e-05, 4.11721296e-04, 1.27362555e-05, 9.83431892e-06,
       1.97038025e-04, 2.34744814e-03, 5.49758552e-04, 1.57057808e-03],
      dtype=float32)
```

بينما تُظهر دالة ()predict من مكتبة sklearn العنوان الأكثر احتمالًا الذي يتنبأ به المُصنِّف، تُظهر دالة ()predict في مكتبة Keras احتمالات كل العناوين المُرشَّحة. في هذه الحالة، يمكن استخدام دالة ()np.argmax لإظهار مؤشر العنوان الأكثر احتمالًا.

```
# index of the class with the highest predicted probability.
print(np.argmax(pred[0]))
# name of this class
print(classes[np.argmax(pred[0])])
# uses axis=1 to find the index of the max value per row
accuracy_score(y_test_num,np.argmax(pred, axis=1))
```

```
1
Duck
0.7529021558872305
```

تحقق هذه الشبكة العصبية البسيطة دقة تبلغ حوالي 75%، وهي دقة مشابهة لدقة SGDClassifier، ولكن ميزة المعماريات العصبية تنبع من براعتها، وهو ما يسمح لك بتجربة معماريات مُختلفة للعثور على أفضل ما يناسب مجموعة بياناتك. تم تحقيق هذه الدقة من خلال معمارية بسيطة تضمنت طبقة مخفية واحدة تحتوي على مئتي خلية عصبية، وإضافة طبقات إضافية تجعل الشبكة أعمق، بينما تؤدي إضافة المزيد من الخلايا العصبية لكل طبقة إلى جعلها أوسع، ويُعدُّ اختيار عدد الطبقات وعدد الخلايا العصبية لكل طبقة عناصر مهمة لتصميم الشبكة العصبية، ولها تأثير كبير على أدائها، ولكنها ليست الطريقة الوحيدة لتحسين الأداء، وفي بعض الحالات قد يكون استخدام نوع مُختلف من معمارية الشبكة العصبية أكثر فاعلية.

## التنبؤ باستخدام الشبكات العصبية الترشيحية

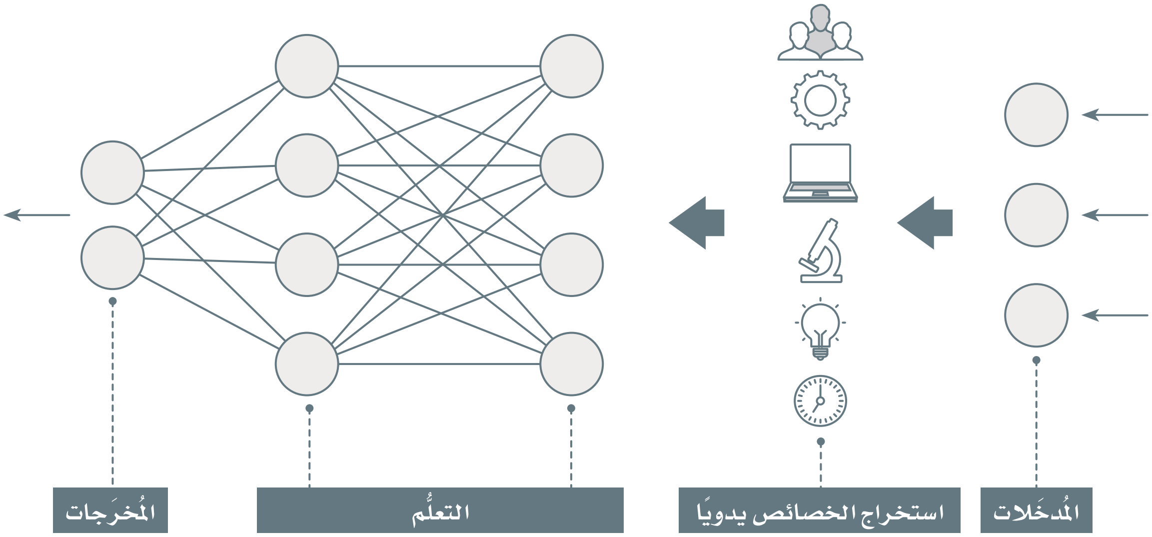
## Prediction Using Convolutional Neural Networks

أحد هذه الأنواع من المعماريات التي تناسب تصنيف الصور بشكلٍ جيّد يتمثّل في **الشبكة العصبية الترشيحية** (Convolutional Neural Network -CNN)، وبما أن الشبكة العصبية الترشيحية تعالج بيانات الإدخال، فإنها تقوم باستمرار بضبط متغيِّرات الفلاتر المرشَّحة لاكتشاف الأنماط بناءً على البيانات التي تراها؛ حتّى تتمكن بشكل أفضل من اكتشاف الخصائص المهمة، ثم تنقل مُخرَجات كل طبقة إلى الطبقة التالية التي يُكتشف فيها خصائص أكثر تعقيدًا إلى أن تُنتج المُخرَجات النهائية.

على الرغم من فوائد الشبكات العصبية المعقدة مثل: الشبكات العصبية الترشيحية إلا أنه من المهم ملاحظة ما يلي:

**الشبكة العصبية الترشيحية (Convolutional Neural Network -CNN):**
هي شبكات عصبية عميقة تتعلّم تلقائيًا تسلسل الخصائص من البيانات الخام مثل الصور، عن طريق تطبيق سلسلة من الفلاتر الترشيحية على بيانات الإدخال، التي يتم تصميمها بحيث تكتشف أنماطًا أو خصائصَ محدَّدة.

- تكمن قوة الشبكات العصبية الترشيحية في قدرتها على أن تستخرج الخصائص المهمة ذات الصلة من الصور بشكل تلقائي، دون الحاجة إلى **هندسة الخصائص اليدوية** (Manual Feature Engineering).
- تحتوي المعماريات العصبية الأكثر تعقيدًا على المزيد من المتغيِّرات التي يجب تعلُّمها من البيانات أثناء التدريب، ويتطلب ذلك مجموعة بيانات تدريب أكبر قد لا تكون متاحة في بعض الحالات، وفي مثل هذه الحالات من غير المحتمل أن يكون إنشاء معمارية معقدة للغاية أمرًا فعَّالًا.
- على الرغم من أن الشبكات العصبية قد حققت بالفعل نتائج مبهرة في معالجة الصور والمهام الأخرى، إلا أنها لا تضمن تقديم أفضل أداء لجميع المشكلات ومجموعات البيانات.
- حتّى لو كانت معمارية الشبكة العصبية أفضل حل ممكن لمُهِمَّة محددة، فقد يستغرق الأمر كثيرًا من الوقت والجهد والموارد الحاسوبية لتجربة خيارات مُختلفة إلى أن يتم العثور على هذه المعمارية. لذلك من الأفضل البدء بنماذج أبسط (لكنها لا تزال فعّالة)، مثل: نموذج SGDClassifier وغيره من النماذج الأخرى الكثيرة المتوفرة في المكتبات مثل: مكتبة sklearn، وبمجرد حصولك على تنبُّؤ أفضل لمجموعة البيانات ووصولك إلى النقطة التي لا يمكن فيها تحسين هذه النماذج أكثر من ذلك، فإن التجريب على المعماريات العصبية الأخرى يُعدُّ خطوة ممتازة.

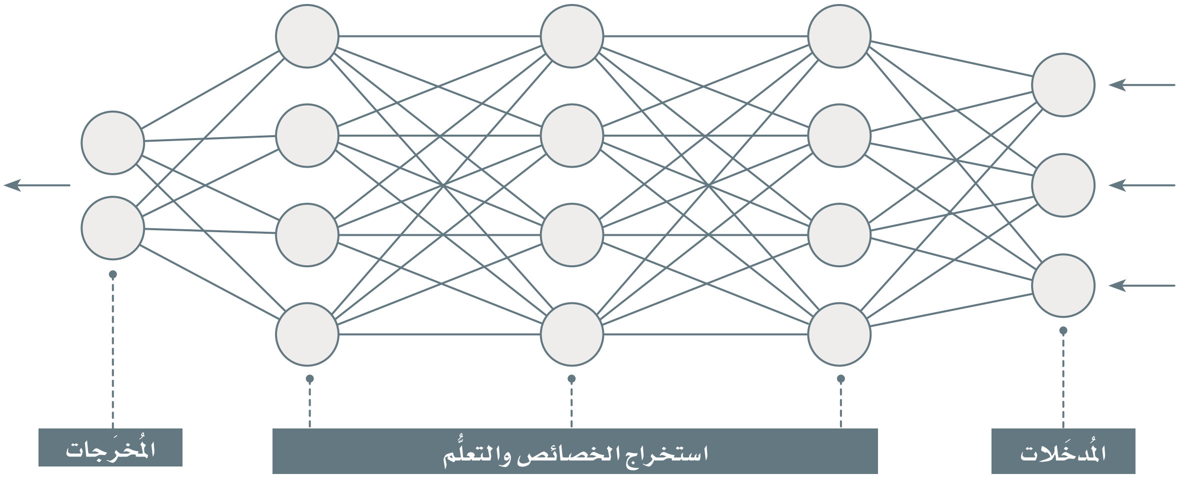



شكل 4.13: شبكة عصبية ذات هندسة خصائص يدوية

**معلومة**

من المزايا الأساسية للشبكات العصبية الترشيحية أنها جيدة جدًا في التعلُّم من كميات كبيرة من البيانات، ويمكنها في العادة أن تحقق مستويات عليا في دقة المهام مثل: تصنيف الصور دون الحاجة إلى هندسة الخصائص اليدوية مثل: المُخطَّط التكراري للتدرجات الموجَّهة.

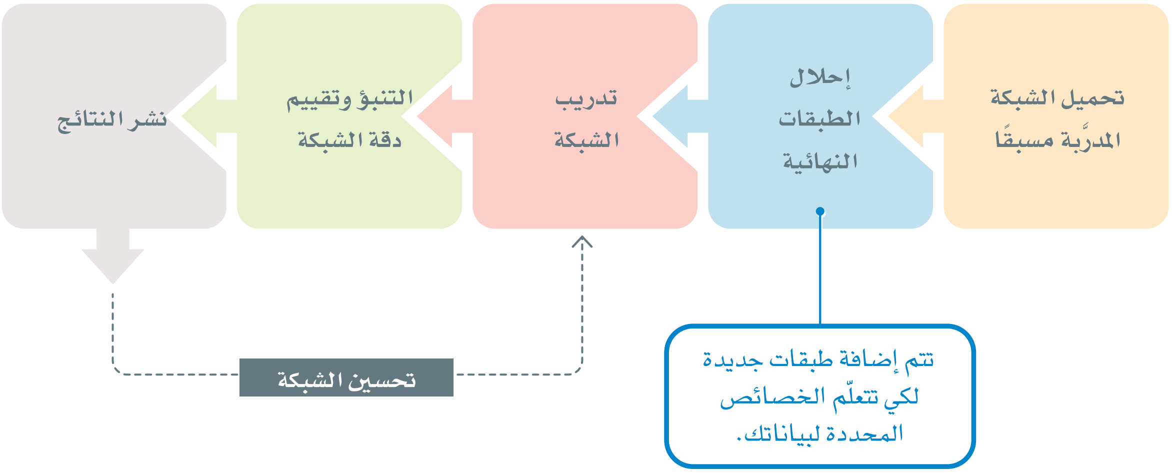

وزارة التعليم
Ministry of Education
2025 - 1447




شكل 4.14: شبكة عصبية ترشيحية من دون هندسة الخصائص اليدوية

## التعلُّم المنقول Transfer Learning

التعلُّم المنقول هو عملية يُعاد فيها استخدام شبكة عصبية مُدرَّبة مسبقًا في حل مُهِمَّة جديدة. في سياق الشبكات العصبية الترشيحية يتضمن التعلُّم المنقول أخذ نموذج مدرَّب مسبقًا على مجموعة بيانات كبيرة وتكييفه على مجموعة بيانات أو مُهِمَّة جديدة، فبدلًا من البدء من نقطة الصفر، يتيح التعلُّم المنقول استخدام النماذج المدرَّبة مسبقًا، أي التي تعلّمت بالفعل خصائص مهمة مثل: الحواف، والأشكال، والنقوش من مجموعة بيانات التدريب.




شكل 4.15: إعادة استخدام الشبكة المدرَّبة مسبقًا


وزارة التعليم
Ministry of Education
2025 - 1447

# تمرينات

**1** ما تحدّيات تصنيف البيانات المرئية؟

**2** لديك مصفوفتا قيم Numpy، وهما مصفوفة X_train ومصفوفة Y_train. كل صف في مصفوفة X_train شكله (3،100، 100) يمثّل صورة بأبعاد 100x100 وبتنسيق RGB. والصف n في المصفوفة Y_train يمثّل تسمية صورة n في مصفوفة X_train. أكمل المقطع البرمجي التالي، بحيث يُسطّح X_train ثم يُدرِّب النموذج MultinomialNB على مجموعة البيانات هذه:

```
from sklearn.naive_bayes import MultinomialNB # imports the Naive Bayes Classifier from sklearn

X_train_flat = np.array(_______________)

model_MNB = MultinomialNB() # new Naive Bayes model

model_MNB.fit(__________________, _________________) # fits model on the flat training data
```

**3** صِف باختصار طريقة عمل الشبكات العصبية الترشيحية وإحدى مميزاتها الرئيسة.

وزارة التعليم
Ministry of Education
2025 - 1447

**4** لديك مصفوفتا قيم Numpy، وهما مصفوفة X_train ومصفوفة Y_train. كل صف في مصفوفة X_train شكله (100،100،3) يمثّل صورة بأبعاد 100x100 وبتنسيق RGB. والصف n في المصفوفة Y_train يمثّل تسمية صورة n في مصفوفة X_train. أكمل المقطع البرمجي التالي، بحيث يطبِّق تحويلات المُخطَّط التكراري للتدرجات الموجَّهة ثم يستخدِم البيانات المحوّلة في تدريب نموذج:

```
from skimage.color import ______________  # used to convert a multi-color (rgb) image to grayscale
from sklearn.______________ import StandardScaler # used to scale the data
from sklearn.naive_bayes import MultinomialNB # imports the Naive Bayes Classifier from sklearn
X_train_gray = np.array([________________(img) for img in X_train]) # converts training data
X_train_hog = ____________________
scaler = StandardScaler()
X_train_hog_scaled = ___________________.fit_transform(X_train_hog)
model_MNB = MultinomialNB()
model_MNB.fit(X_train_flat_scaled, __________________)
```

**5** اذكر بعض عيوب الشبكات العصبية الترشيحية.

________________________________________

________________________________________

________________________________________

________________________________________

________________________________________

________________________________________

________________________________________

________________________________________

________________________________________


وزارة التعليم
Ministry of Education
2025 - 1447

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الثاني

# التعلُّم غير الموجَّه لتحليل الصور

## فهم محتوى الصور

## Understanding Image Content

في سياق رؤية الحاسب يُستخدم التعلُّم غير الموجَّه في مجموعة متنوّعة من المهام مثل: **تقطيع** أو **تجزئة الصورة** (Image Segmentation)، و**تقطيع الفيديو** (Video Segmentation)، و**اكتشاف العناصر الشاذّة** (Anomaly Detection)، ومن الاستخدامات الرئيسة الأخرى للتعلُّم غير الموجَّه: **البحث عن الصورة** (Image Search) ويتضمن البحث في قاعدة بيانات كبيرة من الصور للعثور على الصورة المشابهة للصورة المطلوبة.

تتمثّل الخطوة الأولى لبناء محرك بحث لبيانات صورة في تحديد **دالة التشابه** (Similarity Function) والتي يمكنها تقييم التشابه بين صورتين بناءً على خصائصهما المرئية مثل: الحدود، أو النقش، أو الشكل. وبمجرد أن يُرسِل المستخدِم صورة جديدة ليستعلم عنها، يقوم محرك البحث بالاطلاع على جميع الصور الموجودة في قاعدة البيانات المتاحة، ويعثر على الصور التي بها أعلى درجة تشابه، ويُظهرها للمستخدِم.

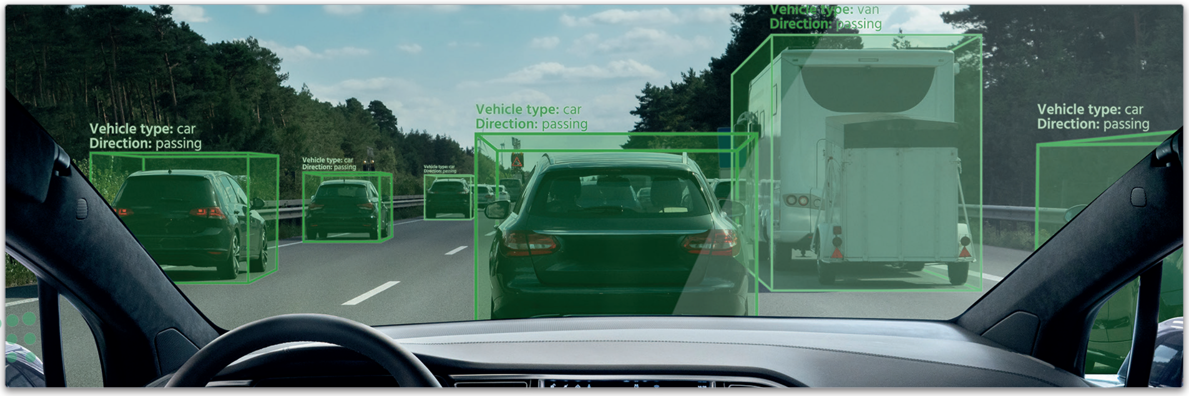
وهناك طريقة بديلة تتمثّل في استخدام دالة التشابه لفصل الصور في عناقيد؛ بحيث يتكون كل عنقود من صور متشابهة بصريًا مع بعضها، ثم يُمَثَّل كل عنقود من خلال **بؤرة تجميع** (Centroid): وهي صورة تقع في مركز العنقود وتمتلك أصغر مسافة عامة (أي اختلاف) من الصور الأخرى في العنقود. وبمجرد أن يُرسِل المستخدِم صورة جديدة للاستعلام عنها، فإن محرك البحث سينتقل إلى جميع العناقيد ويختار العنقود الذي تكون بؤرة تجميعه أكثر تشابهًا مع الصورة المطلوبة من المستخدِم لتظهر له صور العنقود المحددة، ويوضِّح الشكل 4.16 مثالًا على هذا.

### اكتشاف العناصر الشاذّة (Anomaly Detection):

هي عملية تُستخدم لتحديد الأنماط أو الأحداث أو نقاط البيانات الشاذّة أو غير الطبيعية داخل مجموعة البيانات، وتهدف إلى الكشف عن الحالات الغريبة التي تختلف عن المعيار وقد تحتاج إلى استقصاء إضافي.

### تقطيع الصورة (Image Segmentation):

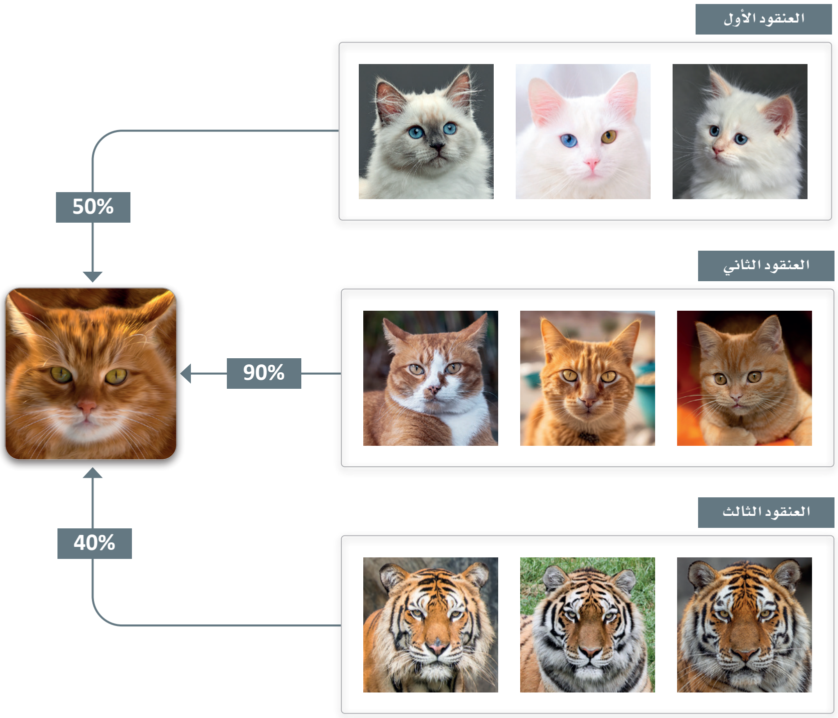
هي عملية تقسيم الصورة إلى أجزاء أو مناطق متعددة تتقاسم خصائص بصرية مشتركة، وتهدف إلى تجزئة الصورة إلى أجزاء مترابطة، وذات مغزى يمكن استخدامها في القيام بتحليل إضافي.




شكل 4.16: رؤية مركبة ذاتية القيادة من خلال تقطيع الصورة



وزارة التعليم
Ministry of Education
2025 - 1447



شكل 4.17: عناقيد التعرّف على الصور

في المثال الموضَّح في الشكل 4.17، تحتوي صورة البحث على تشابه بنسبة: 40% و50% و90% مع بُؤر التجميع لعناقيد الصور الثلاث على التوالي، ويُفترض أن تكون نسبة التشابه بين 0% و 100%، وحصل العنقود الثاني على أعلى نسبة تشابه؛ إذ أنه يشتمل على قطط من نفس سلالة ولون القطّة المحددة في صورة البحث، كما أن نتائج العنقودين الأول والثالث متقاربة (40% و50%)؛ إذ يتشابه العنقودان مع صورة البحث بطرائق مُختلفة، أما العنقود الأول فيتضمن قططًا يختلف نمط ألوانها تمامًا عن المطلوب، وبالرغم من أن العنقود الثالث يمثّل نوعًا مُختلفًا من الحيوانات وهو النمر، فإن نمط اللون مشابه لصورة البحث.

تُشبه عملية تجميع البيانات المرئية في عناقيد، عملية تجميع البيانات الرقمية أو النصيَّة، ومع ذلك تتطلب الطبيعة الفريدة للبيانات المرئية طرائق متخصّصة؛ لتقييم التشابه البصري، وبالرغم من أن الأساليب الأقدم كانت تعتمد على خصائص مصنوعة يدويًا، فقد أدت التطورات الحديثة في التعلُّم العميق إلى تطوير نماذج قوية يمكنها تلقائيًا أن تتعلّم خصائص متطورة من البيانات المرئية غير المُعَنوَنة.

يستخدِم هذا الدرس مُهِمَّة خاصة بتجميع الصور؛ لتوضيح كيف يمكن أن يؤدي استخدام خصائص أكثر تعقيدًا إلى تقديم نتائج أفضل بشكل ملحوظ، وسيوضِّح هذا الدرس -تحديدًا- ثلاث طرائق مُختلفة:

- تسطيح البيانات الأصلية وتجميعها بدون أي هندسة للخصائص.
- تحويل البيانات باستخدام **واصف الخصائص** (Feature Descriptor) الذي يعتمد على **المُخطَّط التكراري للتدرجات الموجَّهة** (HOG) - تعرّفت عليه في الدرس السابق- ثم تجميع البيانات المحوَّلة.
- استخدام نموذج الشبكة العصبية؛ لتجميع البيانات الأصلية في مجموعات عنقودية بدون هندسة الخصائص.

مجموعة بيانات LHI-Animal-Faces (وجوه_الحيوانات) التي استُخدمت في الدرس السابق وستُستخدم في هذا الدرس أيضًا؛ لتقييم التقنيات المتنوّعة لتجميع الصور، وتم تصميم هذه المجموعة في الأصل لمهام التصنيف، وتتضمن العنوان الحقيقي (نوع الحيوان الفعلي) لكل صورة. وفي هذا الدرس، ستُستخدم هذه العناوين فقط للتحقق من صحتها، ولن تُستخدم لتجميع الصور. يجب أن يكون أي أسلوب تجميع أسلوبًا فعّالًا وقادرًا على تجميع الصور مع العنوان نفسه، وفي العنقود نفسه، وعلى فصل الصور ذات العناوين المُختلفة، ووضعها في عناقيد مُتباينة.

## تحميل الصور ومعالجتها أوليًا Loading and Preprocessing Images

يستورد المقطع البرمجي التالي المكتبات التي ستُستخدم لتحميل الصور ومعالجتها أوليًا:

```
%%capture
import matplotlib.pyplot as plt
from os import listdiry

!pip install scikit-image
from skimage.io import imread
from skimage.transform import resize
from skimage import img_as_ubyte

# a palette of 10 colors that will be used to visualize the clusters.
color_palette = ['blue','green','red','yellow','gray','purple','orange',
'pink','black','brown']
```

تقرأ الدالة التالية صور مجموعة بيانات LHI-Animal-Faces (وجوه_الحيوانات) من input_folder (مجلد_المُدخَلات) الخاص بها، وتُعدِّل حجم كل منها بحيث تكون لها أبعاد الطول والعرض نفسها، ثم تقوم بتحسين دالة resize_images() من الدرس السابق بالسماح للمستخدِم بأن يحدِّد قائمة فئات الحيوانات التي يجب أن تؤخذ بالاعتبار، كما أنها تستخدِم سطرًا واحدًا من المقطع البرمجي بلغة البايثون؛ لكي تقرأ كل صورة وتعدِّل حجمها وتخزِّنها:

```
def resize_images_v2(input_folder:str,
                     width:int,
                     height:int,
                     labels_to_keep:list
                    ):
    labels = []           # a list with the label for each image
    resized_images = []   # a list of resized images in np array format
    filenames = []        # a list of the original image file names

    for subfolder in listdir(input_folder):

        print(subfolder)
        path = input_folder + '/' + subfolder

        for file in listdir(path):

            label=subfolder[:-4]  # uses the subfolder name without the "Head" suffix
            if label not in labels_to_keep: continue
            labels.append(label) # appends the label
            #loads, resizes, preprocesses, and stores the image.
            resized_images.append(img_as_ubyte(resize(imread(path+'/'+file),
(width, height))))
            filenames.append(file)

    return resized_images,labels,filenames
```

وزارة التعليم
Ministry of Education
2025 - 1447

**البيانات غير المُنظّمة** (Unstructured Data) متنوّعة، ويمكن أن تحتاج إلى كثير من الوقت والموارد الحاسوبية، ويُعدُّ هذا صحيحًا بشكلٍ خاص عند معالجتها عن طريق أساليب تعلُّم عميقة ومعقدة، كما سيُنفذ لاحقًا في هذا الدرس، ولتقليل الوقت الحسابي يتم تطبيق دالة resize_images_v2() على مجموعة فرعية من الصور من فئات الحيوانات:

```
resized_images,labels,filenames=resize_images_v2(
            "AnimalFace/Image",
            width = 224,
            height = 224,
            labels_to_keep=['Lion', 'Chicken', 'Duck', 'Rabbit', 'Deer',
'Cat', 'Wolf', 'Bear', 'Pigeon', 'Eagle']
            )
```

```
BearHead          MonkeyHead
CatHead           Natural
ChickenHead       PandaHead
CowHead           PigeonHead
DeerHead          RabbitHead
DuckHead          SheepHead
EagleHead         TigerHead
ElephantHead      WolfHead
LionHead
```

هذه العناوين العشرة التي سيتم استخدامها.

يمكنك بسهولة تعديل المتغيِّر labels_to_keep (العناوين_المحتفظ بها)؛ للتركيز على فئات معيّنة، وستلاحظ أن عرض الصور وارتفاعها تم ضبطهما على 224 × 224، بدلًا من الشكل 100 × 100 الذي استُخدم في الدرس السابق؛ لأن إحدى طرائق التجميع القائمة على التعلُّم العميق -الواردة في هذا الدرس- تتطلب أن تكون للصور هذه الأبعاد، ولذا اعتُمد الشكل 224 × 224؛ لضمان منح حق الوصول لجميع الطرائق إلى المُدخَلات نفسها.

كما ذُكر في الدرس السابق فإن القوائم الأصلية: resized_images (الصور_المُعدَّل حجمها)، وlabels (العناوين)، وfilenames (أسماء الملفات) تشتمل على الصور التي تنتمي لكل فئة مُجمَّعة معًا. على سبيل المثال: تظهر جميع صور Lion (الأسد) معًا في بداية القائمة المُعدَّل حجمها، وقد يُضلل ذلك العديد من الخوارزميات، خاصة في مجال رؤية الحاسب، وطالما أنه يمكن فهرسة الصور عشوائيًا لكل قائمة من القوائم الثلاث، فمن المهم التأكد من استخدام الترتيب العشوائي نفسه لهذه القوائم. وبخلاف ذلك، من المستحيل العثور على العنوان الصحيح لصورة معيّنة أو اسم الملف الصحيح لها.

في الدرس السابق، تم إجراء **إعادة الترتيب** (Shuffling) باستخدام الدالة train_test_split()، وبما أن هذه الدالة غير قابلة للتطبيق على مهام التجميع، فستستخدِم المقطع البرمجي التالي لإعادة الترتيب:

```
import random

#connects the three lists together, so that they are shuffled in the same order
connected = list(zip(resized_images,labels,filenames))
random.shuffle(connected)
# disconnects the three lists
resized_images,labels,filenames= zip(*connected)
```

تتمثّل الخطوة التالية في تحويل قائمتي resized_images (الصور_المُعدَّل حجمها)، وlabels (العناوين) إلى مصفوفات numpy، وكما هو الحال في الدرس السابق يُستخدم الاسمان المتغيِّران القياسيان (X،Y) لتمثيل البيانات والعناوين:

```
import numpy as np # used for numeric computations
X = np.array(resized_images)
Y = np.array(labels)

X.shape
```

```
(1085, 224, 224, 3)
```

يتحقق شكل البيانات من أنها تشمل 1,085 صورة، كل صورة منها ذات أبعاد 224 × 224، وذات ثلاث قنوات ألوان RGB.

## التجميع بدون هندسة الخصائص Clustering without Feature Engineering

ستركز محاولة التجميع الأولى على القيام بتسطيح الصور؛ لتحويل كل منها إلى متَّجَه أحادي البُعد أرقامه 224 × 224 × 3 = 150,528 رقمًا.

وعلى غرار خوارزميات التصنيف التي تم توضيحها في الدرس السابق، فإن معظم خوارزميات التجميع تتطلب هذا النوع من التنسيق المتَّجَهي.

```
X_flat = np.array([img.flatten() for img in X])
X_flat[0].shape
```

```
(150528,)
```

```
X_flat[0] # prints the first flat image
```

```
array([107, 146, 102, ...,  91,  86, 108], dtype=uint8)
```

كل قيمة عددية في هذا التنسيق المسطح ذات قيمة ألوان RGB تتراوح بين 0 و255، وفي الدرس السابق، تمّ توضيح أن التحجيم القياسي والتسوية يؤديان أحيانًا إلى تحسين نتائج بعض خوارزميات التعلُّم الآلي.

يمكن استخدام المقطع البرمجي التالي لتسوية القيم وجعلها ما بين 0 و1:

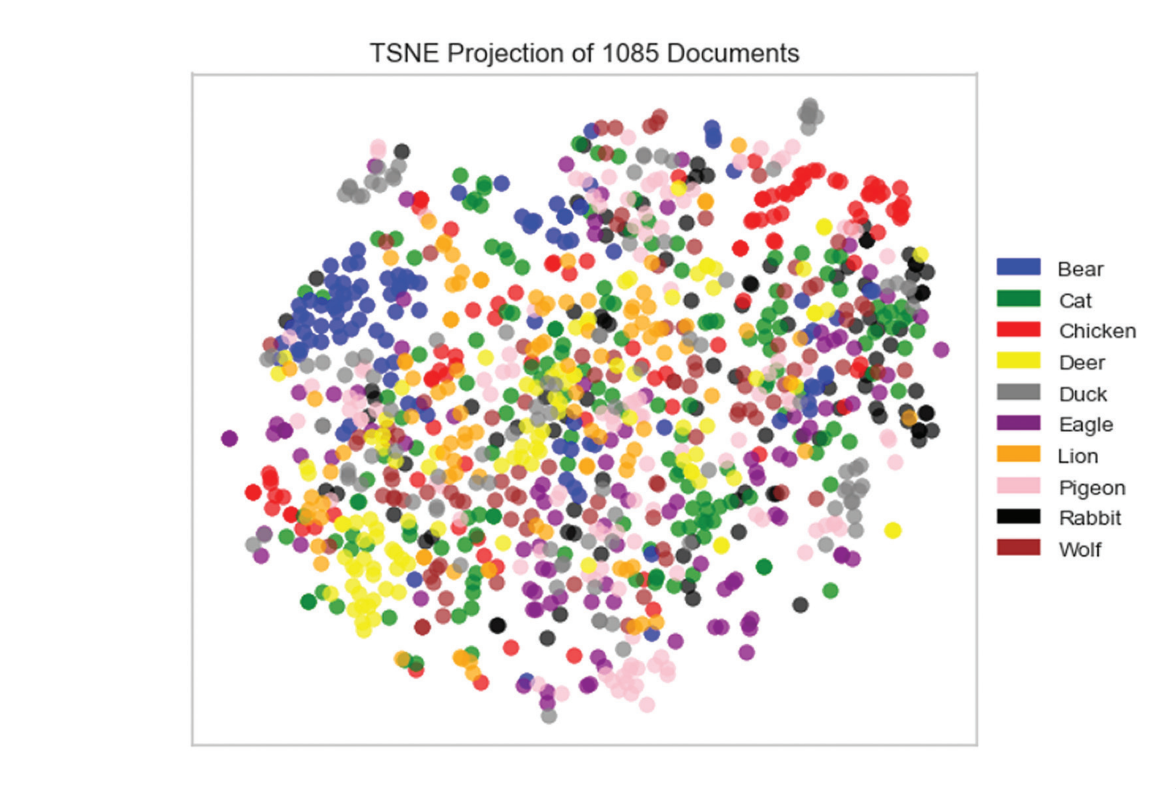
```
X_norm = X_flat / 255
X_norm[0]
```

```
array([0.41960784, 0.57254902, 0.4       , ..., 0.35686275, 0.3372549 ,
       0.42352941])
```

يمكـن الآن تصويـر البيانـات بصريًـا باسـتخدام أداة TSNEVisualizer المألوفـة مـن مكتبـة yellowbrick، وتم استخدام هذه الأداة أيضًا في الدرس الثاني من الوحدة الثالثة؛ لتصوير العناقيد بصريًا في البيانات النصيَّة.

```
%%capture
!pip install yellowbrick
from yellowbrick.text import TSNEVisualizer
```

```
tsne = TSNEVisualizer(colors = color_palette) # initializes the tool
tsne.fit(X_norm, y) # uses TSNE to reduce the data to 2 dimensions
tsne.show();
```



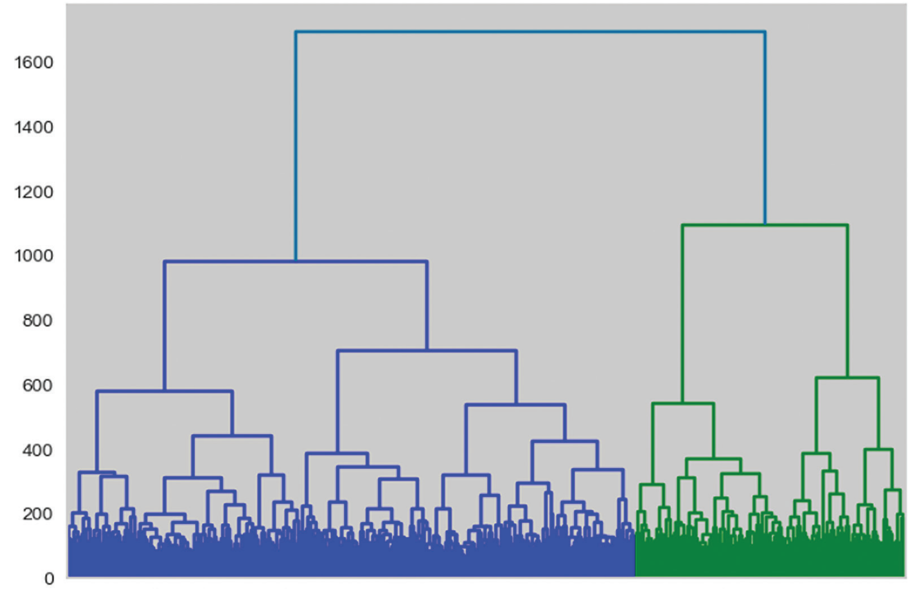

شكل 4.18: تصوير العناقيد

التصوير التمهيدي هـذا ليس كما هـو متوقَّع، فيبدو أن فئـات الحيوانات المُختلفـة مختلطة ببعضهـا، دون تمييز واضح بينهـا ودون عناقيـد واضحـة لهـا، ويدل ذلـك على أن مجـرد القيـام بتسـطيح بيانـات الصـورة الأصلية من المحتمل ألا يؤدي إلى نتائج ذات جودة عالية.

بعـد ذلـك، ستُسـتخدم خوارزميـة **التجميـع التكتلي** (Agglomerative Clustering) نفسـها التـي اسـتُخدمت في الدرس الثـاني مـن الوحـدة الثالثـة؛ لتجميـع البيانـات في متغيِّر X_norm، ويسـتورد المقطـع البرمجـي التـالي مجموعة الأدوات المطلوبة، ويصوِّر الرسم الشجري لمجموعة البيانات:

```python
from sklearn.cluster import AgglomerativeClustering # used for agglomerative clustering
import scipy.cluster.hierarchy as hierarchy

hierarchy.set_link_color_palette(color_palette) # sets the color palette
plt.figure()

# iteratively merges points and clusters until all points belong to a single cluster
linkage_flat = hierarchy.linkage(X_norm, method = 'ward')
hierarchy.dendrogram(linkage_flat)
plt.show()
```

ward (وارد) عبارة عن طريقة ربط تُستخدم في التجميع التكتلي الهرمي.




شكل 4.19: الرسم الشجري يُصنف البيانات إلى عنقودين

يكشف الرسم الشجري عنقودين كبيرين يمكن تقسيمهما إلى عناقيد أصغر، ويُستخدم المقطع البرمجي التالي أداة AgglomerativeClustering (التجميع التكتلي)؛ لإنشاء عشرة عناقيد، وهو العدد الفعلي للعناقيد الموجودة في البيانات:

```python
AC = AgglomerativeClustering(linkage = 'ward',n_clusters = 10)
AC.fit(X_norm) # applies the tool to the data

pred = AC.labels_ # gets the cluster labels

pred
```

```
array([9, 6, 3, ..., 4, 4, 3], dtype=int64)
```

وأخيرًا، تُستخدم مؤشرات: Homogeneity (التجانس)، وCompleteness (الاكتمال)، وAdjusted Rand (راند المُعدَّل) وكلها تعرّفت عليها في الدرس الثاني من الوحدة الثالثة؛ لتقييم جودة العناقيد الناتجة.

وزارة التعليم
Ministry of Education
2025 - 1447

```
from sklearn.metrics import homogeneity_score, adjusted_rand_score,
completeness_score

print('\nHomogeneity score:', homogeneity_score(y, pred))
print('\nAdjusted Rand score:', adjusted_rand_score(y, pred))
print('\nCompleteness score:', completeness_score(y, pred))
```

```
Homogeneity score: 0.09868725008128477

Adjusted Rand score: 0.038254515908926826

Completeness score: 0.101897123096584
```

كما سبق توضيحه بالتفصيل في الدرس الثاني من الوحدة الثالثة، فإن مؤشري التجانس والاكتمال يأخذان قيمًا بين 0 و1، وترتفع قيمة مؤشر التجانس إلى أقصى حد عندما يكون لجميع نقاط العنقود الواحد العنوان الحقيقي الأساسي نفسه، كما ترتفع قيمة مؤشر الاكتمال إلى الحد الأقصى عندما تنتمي جميع نقاط البيانات التي تحمل العنوان الحقيقي الأساسي نفسه إلى العنقود نفسه، وأخيرًا يأخذ مؤشر راند المُعدَّل قيمًا بين 0.5- و1.0، وترتفع إلى الحد الأقصى عندما تكون جميع نقاط البيانات التي لها العنوان نفسه في العنقود نفسه، وتكون جميع النقاط ذات العناوين المُختلفة في عناقيد متباينة، وكما هو متوقَّع تفشل الخوارزمية بعد تصوير البيانات في العثور على عناقيد عالية الجودة تتطابق مع فئات الحيوانات الفعلية، حيث أن قيم المؤشرات الثلاث منخفضة للغاية، وعلى الرغم من أن مجرد القيام بتسطيح البيانات كان كافيًا للحصول على نتائج معقولة لتصنيف الصور، إلا أن تجميع الصور في عناقيد يُمثِّل مشكلة أكثر صعوبة.

## التجميع بانتقاء الخصائص Clustering with Feature Selection

في الدرس السابق تم توضيح أنّ استخدام تحويل **المُخطَّط التكراري للتدرجات الموجَّهة** (HOG) لتحويل بيانات الصور إلى صيغة أكثر دلالة يؤدي إلى إنجاز أعلى بشكل ملحوظ في تصنيف الصور، وسيُطبَّق التحويل نفسه لاختبار ما إذا كان بإمكانه أيضًا تحسين نتائج مهام تجميع الصور.

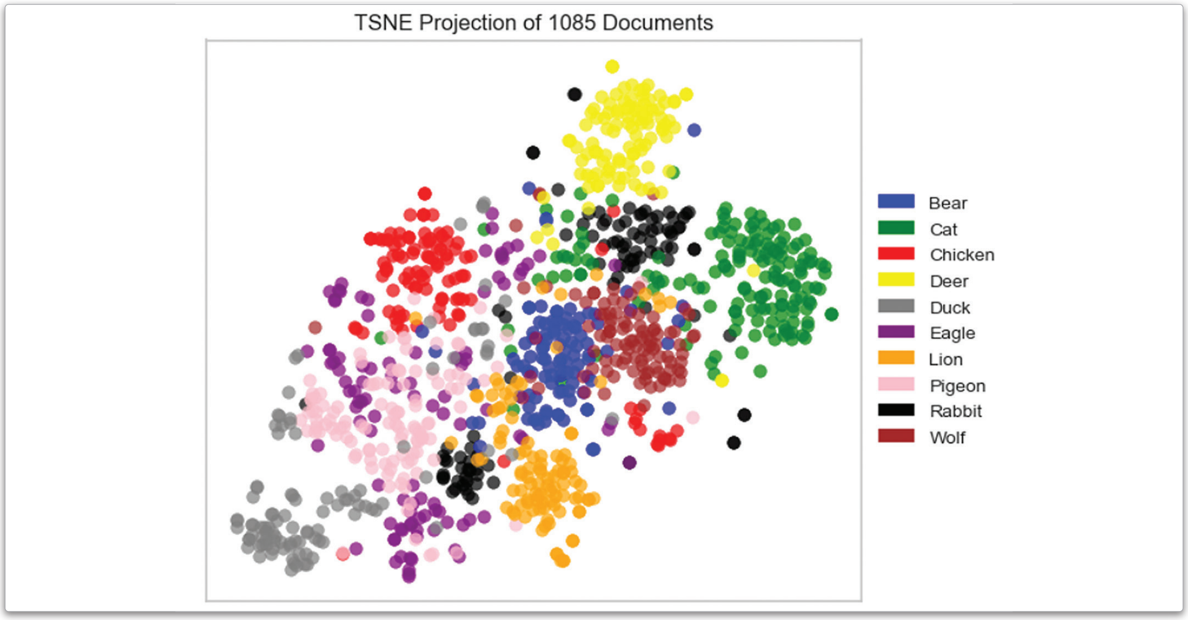
```
from skimage.color import rgb2gray
from skimage.feature import hog
# converts the list of resized images to an array of grayscale images
X_gray = np.array([rgb2gray(img) for img in resized_images])
# computes the HOG features for each grayscale image in the array
X_hog = np.array([hog(img) for img in X_gray])
X_hog.shape
```

```
(1085, 54756)
```

يكشف شكل البيانات المحوَّلة أن كل صورة تُمثَّل الآن على هيئة متَّجَه بقيمة عددية هي: أربعة وخمسون ألفًا وسبعمئة وستة وخمسون (54,756).

يستخدِم المقطع البرمجي التالي أداة TSNEVisualizer لتصوير هذا التنسيق الجديد:

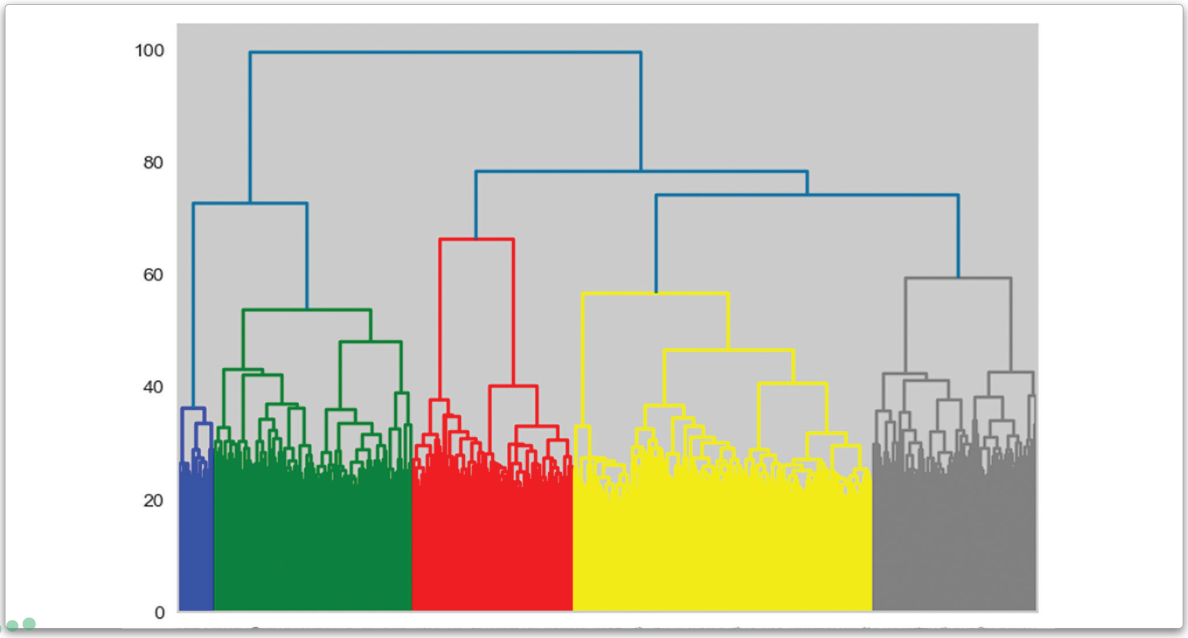
```
tsne = TSNEVisualizer(colors = color_palette)
tsne.fit(X_hog, y)
tsne.show();
```




شكل 4.20: تصوير العناقيد

يُعدُّ هذا التصوير أكثر مصداقية من الذي تم إنتاجه للبيانات غير المحوَّلة، وعلى الرغم من وجود بعض الشوائب، فإن الشكل يُظهر عناقيد واضحة ومفصولة جيدًا، ويمكن الآن حساب الرسم الشجري لمجموعة البيانات هذه.

```
plt.figure()
linkage_2 = hierarchy.linkage(X_hog,method = 'ward')
hierarchy.dendrogram(linkage_2)
plt.show()
```




شكل 4.21: الرسم الشجري لفئات وجوه الحيوانات المُختلفة باستخدام مُخطَّط تكراري للتدرجات الموجَّهة (HOG)

يقترح الرسم الشجري خمسة عناقيد، وهو بالضبط نصف العدد الصحيح البالغ عشرة عناقيد. يتبنى المقطع البرمجي التالي هذا الاقتراح ويطبِّق أداة AgglomerativeClustering (التجميع التكتلي) ويُظهر نتائج المؤشرات الثلاثة:

```
AC = AgglomerativeClustering(linkage = 'ward', n_clusters = 5)
AC.fit(X_hog)
pred = AC.labels_

print('\nHomogeneity score:', homogeneity_score(y, pred))
print('\nAdjusted Rand score:', adjusted_rand_score(y, pred))
print('\nCompleteness score:', completeness_score(y, pred))
```

```
Homogeneity score: 0.4046340612330986

Adjusted Rand score: 0.29990205334627734

Completeness score: 0.6306921317302154
```

تكشف النتائج أنه على الرغم من أن عدد العناقيد التي تم استخدامها كان أقل بكثير من العدد الصحيح، إلا أن النتائج أفضل بكثير من النتائج التي ظهرت عند استخدام الرقم الصحيح على البيانات غير المحوَّلة.

ويوضِّح ذلك ذكاء التحويل بواسطة المُخطَّط التكراري للتدرجات الموجَّهة، ويُثبت أنه يمكن أن يؤدي إلى تحسينات رائعة في الأداء لكل من مهام التعلُّم الموجَّه ومهام التعلُّم غير الموجَّه في رؤية الحاسب، ولإكمال التحليل يُعيد المقطع البرمجي التالي تجميع البيانات المحوَّلة بالعدد الصحيح للعناقيد:

```
AC = AgglomerativeClustering(linkage = 'ward', n_clusters = 10)
AC.fit(X_hog)
pred = AC.labels_

print('\nHomogeneity score:', homogeneity_score(y, pred))
print('\nAdjusted Rand score:', adjusted_rand_score(y, pred))
print('\nCompleteness score:', completeness_score(y, pred))
```

```
Homogeneity score: 0.5720932612704411

Adjusted Rand score: 0.41243540297103065

Completeness score: 0.617016965322667
```

وكما هو متوقَّع، زادت قيم المؤشرات بشكل عام، فعلى سبيل المثال تجاوز كل من التجانس والاكتمال الآن 0.55، مما يدل على أن الخوارزمية تقوم بعمل أفضل فيما يتعلق بكل من: وضع الحيوانات التي تنتمي لفئة واحدة في العنقود نفسه، وإنشاء عناقيد **نقية** (Pure) تتكون في الغالب من فئة الحيوان نفسه.

## التجميع باستخدام الشبكات العصبية Clustering Using Neural Networks

أحدث استخدام نماذج التعلُّم العميق (الشبكات العصبية العميقة ذات الطبقات المتعددة) ثورة في مجال تجميع الصور من خلال توفير خوارزميات قوية وعالية الدقة، ويمكنها تجميع الصور المتشابهة معًا تلقائيًا دون الحاجة إلى هندسة الخصائص. تعتمد العديد من الطرائق التقليدية لتجميع الصور على خاصية **المستخرجات** (Extractors) لاستخراج معلومات ذات مغزى من صورة ما، واستخدام هذه المعلومات لتجميع الصور المتشابهة معًا، ويمكن أن تستغرق هذه العملية وقتًا طويلًا وتتطلب خبرة في المجال لتصميم خاصية المستخرِجات بخصائص فعّالة. بالإضافة إلى ذلك -وكما تم التوضيح في الدرس السابق- على الرغم من أن خاصية **الواصِفات** (Descriptors) مثل: تحويل المُخطَّط التكراري للتدرجات الموجَّهة يمكنها بالفعل تحسين النتائج، إلا أنها بعيدة كل البُعد عن الكمال، وبالتأكيد يوجد مجال للتحسين. من ناحية أخرى، يتمتع التعلُّم العميق بالقدرة على تعلُّم تمثيلات الخصائص من البيانات الخام تلقائيًا، ويتيح ذلك لطرائق التعلُّم العميق معرفة الخصائص شديدة التمايز التي تلتقط الأنماط الهامة وراء البيانات، مما يؤدي إلى تجميع أكثر دقة وقوة، ولتحقيق ذلك تُستخدم عدة طبقات مُختلفة في الشبكة العصبية بما فيها:

- **الطبقات الكثيفة** (Dense Layers)
- **طبقات التجميع** (Pooling Layers)
- **طبقات الإقصاء** (Dropout Layers)

في الشبكة العصبية في الدرس الأول من الوحدة الثالثة، تم استخدام طبقة مخفية مكونة من ثلاث مئة خلية عصبية من نموذج **الكلمة إلى المتَّجَه** (Word2Vec)؛ لتمثيل كل كلمة، وفي تلك الحالة دُرِّب نموذج الكلمة إلى المتَّجَه مسبقًا على مجموعة بيانات كبيرة جدًا تحتوي على ملايين الأخبار من **أخبار قوقل** (Google News). تُعدُّ نماذج الشبكات العصبية المدرَّبة مسبقًا شائعة أيضًا في مجال رؤية الحاسب، ومن الأمثلة المعهودة على ذلك نموذج VGG16 الذي يشيع استخدامه في مهام التعرّف على الصور، ويتبع نموذج VGG16 معمارية عميقة قائمة على الشبكات العصبية الترشيحية يوجد بها ست عشرة طبقة، ويُعدُّ نموذجًا موجَّهًا دُرِّب على مجموعة بيانات كبيرة من الصور المُعَنوَنة تسمى **شبكة الصور** (ImageNet)، ومع ذلك، تتكون مجموعة بيانات التدريب الخاصة بنموذج VGG16 من ملايين الصور ومئات العناوين المُختلفة، مما يحسِّن بشكل كبير من قدرة النموذج على فهم الأجزاء المُختلفة من الصورة، وعلى غرار الشبكة العصبية الترشيحية البسيطة الموضَّحة في الشكل 4.22، ويستخدِم نموذج VGG16 أيضًا طبقة كثيفة نهائية تحتوي على أربعة آلاف وستة وتسعين خلية عصبية لتمثيل كل صورة قبل إدخالها في **طبقة المُخرَج** (Output Layer)، ويوضِّح هذا القسم كيف يمكن تكييف نموذج VGG16 لتجميع الصور، على الرغم من أنه صُمِّم في الأصل لتصنيف الصور:

1. حمِّل النموذج VGG16 الذي دُرِّب مسبقًا.
2. احذف طبقة المُخرَج من النموذج، فذلك يجعل الطبقة الأخيرة الكثيفة هي طبقة المُخرَج الجديدة.
3. استخدم **النموذج المقتطع** (Truncated Model) - النموذج السابق الذي اقتُطعت الطبقة الأخيرة منه-؛ لتحويل كل صورة في مجموعة بيانات Animal Faces (وجوه الحيوانات) إلى متَّجَه عددي له أربعُ آلاف وستٌ وتسعون قيمة.
4. استخدم التجميع التكتلي؛ لتجميع المتَّجَهات الناتجة عن ذلك.

**الطبقة الكثيفة (Dense Layer):**

هي طبقة في الشبكات العصبية ترتبط فيها كل العُقد التي في الطبقة السابقة بكل العُقد التي في الطبقة الحالية، حيث يتم تمرير الإشارات من العُقد في الطبقة السابقة في الشبكة إلى العُقد في الطبقة الحالية بواسطة وزنية محدَّدة، وتُطبَّق **دالَّة التنشيط** (Activation Function) على الإشارات المرسَلة إلى الطبقة الكثيفة لتوليد نتائج الإخراج النهائية.

**طبقة التجميع (Pooling Layer):**

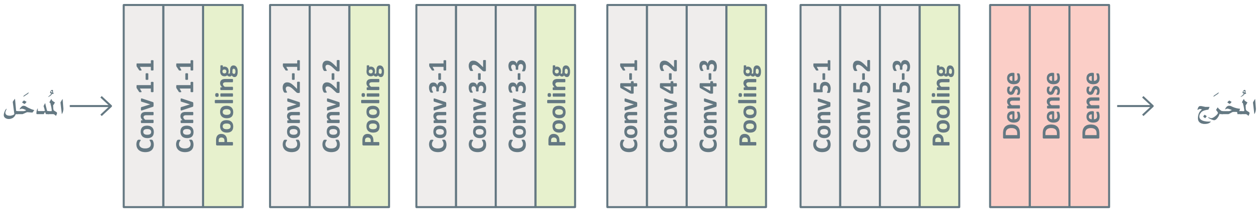
هي طبقة في الشبكات العصبية تُستخدم لتقليل الأبعاد الفراغية لبيانات المُدخَلات.

**طبقة الإقصاء (Dropout Layer):**

هي طريقة تنظيم تُستخدم لمنع فرط التخصيص في نموذج لمجموعة بيانات في الشبكات العصبية عن طريق إقصاء عُقد موجودة في الطبقة خلال كلّ دورة تدريب.



وزارة التعليم
Ministry of Education
2025 - 1447




شكل 4.22: معمارية نموذج VGG16

يمكن استخدام مكتبة TensorFlow ومكتبة Keras اللتين تعرّفت عليهما في الدرس السابق للوصول إلى نموذج VGG16 واقتطاعه، وتتمثّل الخطوة الأولى في استيراد جميع الأدوات المطلوبة:

```
from keras.applications.vgg16 import VGG16  # used to access the pre-trained VGG16 model
from keras.models import Model

model = VGG16() # loads the pretrained VGG16 model
# removes the output layer
model = Model(inputs = model.inputs, outputs = model.layers[-2].output)
```

يحذف الطبقة الأخيرة من المُخرَج.

يطبِّق المقطع البرمجي التالي المعالجة الأولية الأساسية نفسها التي يتطلبها نموذج VGG16 مثل: تحجيم قيم ألوان RGB لتكون بين 0 و1:

```
from keras.applications.vgg16 import preprocess_input
X_prep = preprocess_input(X)
X_prep.shape
```

```
(1085, 224, 224, 3)
```

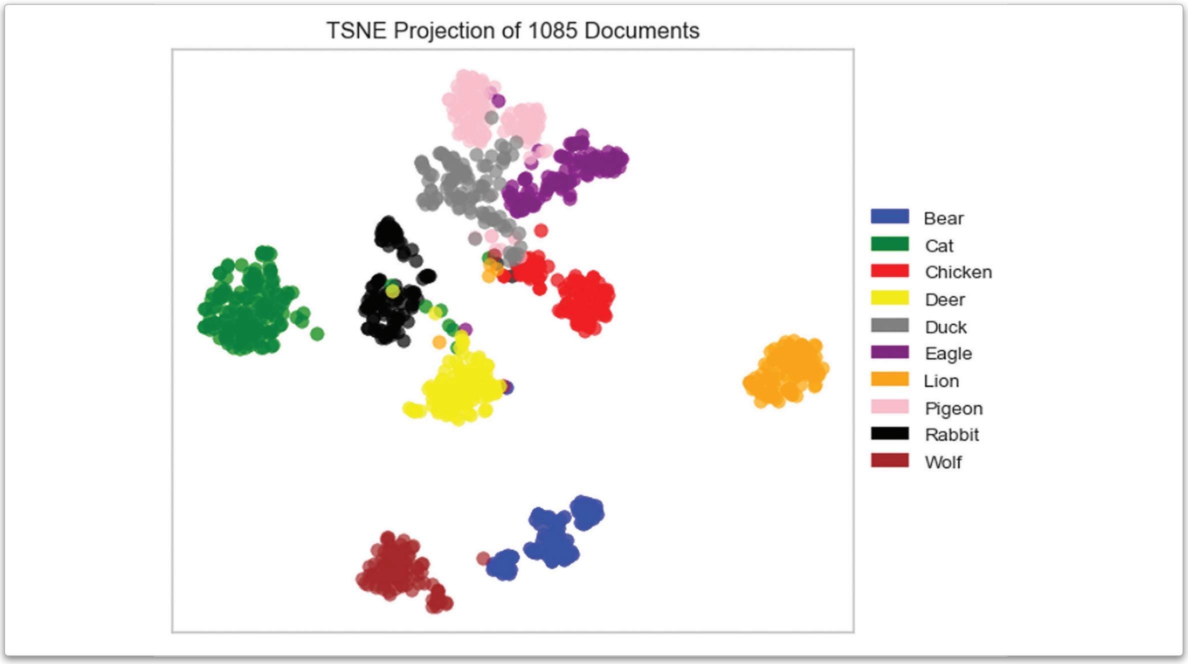
لاحظ أن شكل البيانات يظل كما هو، أي: ألفٌ وخمسٌ وثمانون صورة، كل صورة منها أبعادها 224 × 224، وثلاث قنوات ألوان RGB، وبعد ذلك يمكن استخدام النموذج المقتطع لتحويل كل صورة إلى متَّجه مكوّن من 4,096 عدد.

```
X_VGG16 = model.predict(X_prep, use_multiprocessing = True)
X_VGG16.shape
```

```
34/34 [==============================] - 57s 2s/step

(1085, 4096)
```

يُضبط متغيِّر المعالجة المتعددة multiprocessing=True (تفعيل المعالجة المتعددة) لتسريع العملية من خلال حساب المتَّجهات للصور المتعددة بالتوازي، وقبل إكمال خطوة التجميع يُستخدم المقطع البرمجي التالي لتصوير **البيانات المتّجِهة** (Vectorized Data):

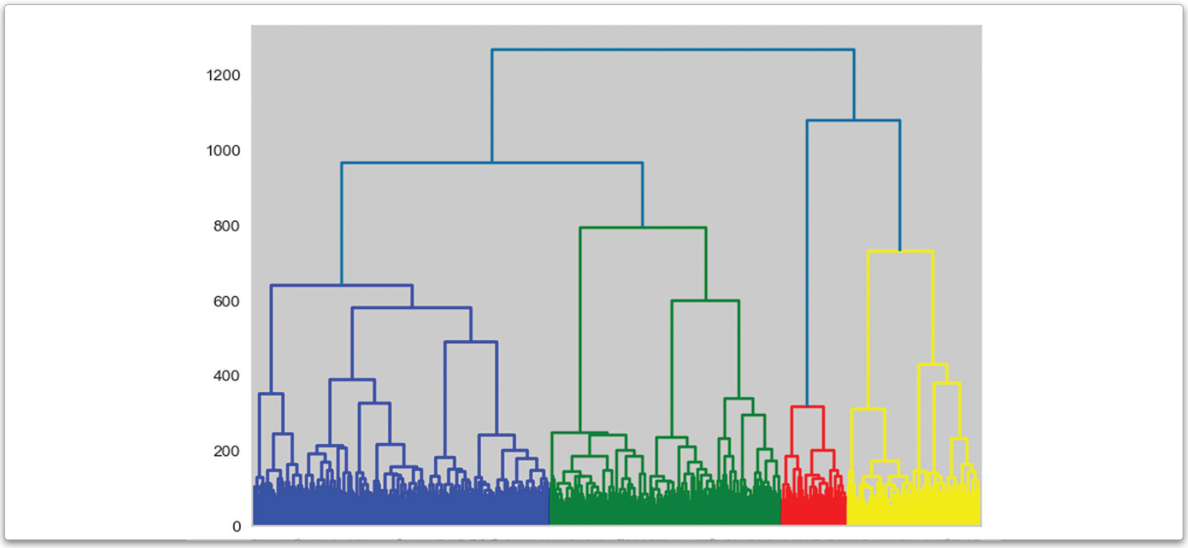
```
tsne = TSNEVisualizer(colors = color_palette)
tsne.fit(X_VGG16, labels)
tsne.show();
```




شكل 4.23: تصوير العناقيد المتشابهة

تُعدُّ النتائج مذهلة؛ لأن التصوير الجديد يكشف عناقيد مفصولة عن بعضها بوضوح وتكاد تكون كاملة، كما أن الفصل هنا أفضل بكثير من الفصل الذي كان في البيانات التي حُوِّلت بواسطة المُخطَّط التكراري للتدرجات الموجَّهة.

```
linkage_3 = hierarchy.linkage(X_VGG16, method = 'ward')
plt.figure()
hierarchy.dendrogram(linkage_3)
plt.show()
```




شكل 4.24: الرسم الشجري الهرمي لفئات وجوه الحيوانات المُختلفة باستخدام نموذج VGG16

يقترح الرسم الشجري أربعة عناقيد، وفي هذه الحالة يمكن للممارس أن يتجاهل الاقتراح بسهولة، ويتّبع التصوير السابق بدلًا منه والذي يبيّن بوضوح وجود عشرة عناقيد.

وزارة التعليم
Ministry of Education
2025 - 1447

يستخدِم المقطع البرمجي التالي التجميع التكتلي ويوضِّح قيم المؤشرات لكل من العناقيد الأربعة والعناقيد العشرة:

```
AC = AgglomerativeClustering(linkage = 'ward',n_clusters = 4)
AC.fit(X_VGG16)
pred=AC.labels_

print('\nHomogeneity score:', homogeneity_score(y, pred))
print('\nAdjusted Rand score:', adjusted_rand_score(y, pred))
print('\nCompleteness score:', completeness_score(y, pred))
```

```
Homogeneity score: 0.504687456015823

Adjusted Rand score: 0.37265351562538257

Completeness score: 0.9193141240200559
```

```
AC = AgglomerativeClustering(linkage='ward',n_clusters = 10)
AC.fit(X_VGG16)
pred=AC.labels_

print('\nHomogeneity score:', homogeneity_score(y, pred))
print('\nAdjusted Rand score:', adjusted_rand_score(y, pred))
print('\nCompleteness score:', completeness_score(y, pred))
```

```
Homogeneity score: 0.8403973102506642

Adjusted Rand score: 0.766734821176714

Completeness score: 0.8509145102288217
```

تثبت النتائج صحة الأدلة التي قدمها التصوير، وتؤدي التحولات التي أنتجها نموذج VGG16 إلى نتائج مذهلة إلى حد كبير لكل من العناقيد الأربعة والعناقيد العشرة. في الواقع، ظهرت نتائج شبه مثالية لجميع المؤشرات الثلاثة عند استخدام عشرة عناقيد، مما يثبت أن النتائج غالبًا تتوافق تمامًا مع فئات الحيوانات في مجموعة البيانات.

يُعدُّ نموذج VGG16 من أقدم نماذج الشبكات العصبية الترشيحية عالية الذكاء المدرَّبة مسبقًا لغرض استخدامها في تطبيقات رؤية الحاسب، ومع ذلك نُشرت العديد من نماذج الشبكات العصبية الترشيحية الذكية الأخرى المدرَّبة مسبقًا والتي تجاوز أداؤها أداء نموذج VGG16.


وزارة التعليم
Ministry of Education
2025 - 1447

# تمرينات

**1** اذكر الميزة التي تتمتع بها تقنيات التعلُّم غير الموجَّه مقارنة بتقنيات التعلُّم الموجَّه في تحليل الصور.

---

**2** لديك مصفوفة قيم موحدة X_flat تشمل صورًا مُسطحة، وكل صف في المصفوفة يمثِّل صورة مسطحة مُختلفة على هيئة متتالية من الأعداد الصحيحة تتراوح بين 0 و255. أكمل المقطع البرمجي التالي، بحيث يستخدم التجميع التكتلي في تصنيف الصور التي من X_flat إلى خمسة عناقيد مُختلفة:

```
from ______________ import AgglomerativeClustering # used for agglomerative clustering

AC = AgglomerativeClustering(linkage='ward', ______________)

X_norm = _________________ # normalizes the data

AC.fit(X_norm) # applies the tool to the data

pred = AC._________________ # gets the cluster labels
```

---

**3** عدِّد بعض مزايا استخدام التعلُّم العميق التي يمتاز بها على طرائق تجميع الصور التقليدية.

**4** لديك مصفوفة قيم موحدة X_flat تشمل صورًا مسطحة، وكل صف في المصفوفة يمثِّل صورة مسطحة مُختلفة على هيئة متتالية من الأعداد الصحيحة تتراوح بين 0 و255. أكمل المقطع البرمجي التالي، بحيث يستخدِم طريقة وارد (ward) لإنشاء وتصوير رسم شجري للصور في هذه المصفوفة:

```
import scipy.cluster.hierarchy as hierarchy # visualizes and supports hierarchical clustering tasks

import ____________________ as plt

X_norm = __________________ # normalizes the data

plt.figure() # creates a new empty figure

linkage_flat=hierarchy.linkage(________________, method='_________________')

hierarchy._________________(linkage_flat)

plt.show() #shows the figure
```

**5** صِف الطريقة التي يُطبَّق بها التجميع بالشبكات العصبية في تحليل الصور.

____________________________________________________________

____________________________________________________________

____________________________________________________________

____________________________________________________________

____________________________________________________________

____________________________________________________________

____________________________________________________________

____________________________________________________________

____________________________________________________________

____________________________________________________________

____________________________________________________________



وزارة التعليم
Ministry of Education
2025 - 1447

رابط الدرس الرقمي

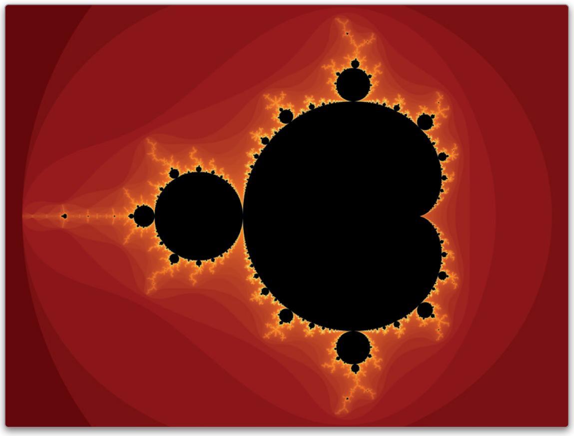
www.ien.edu.sa

الدرس الثالث

# توليد البيانات المرئية

## استخدام الذكاء الاصطناعي في توليد الصور Using AI to Generate Images

بينما ركزت خوارزميات رؤية الحاسب التي تم توضيحها في الدرسين السابقين من هذه الوحدة على فهم الجوانب المُختلفة لصورة معيّنة، يُركّز مجال **توليد الصور** (Image Generation) في هذا الدرس على إنشاء صور جديدة. فمجال **توليد الصور** (Image Generation) له تاريخ طويل يعود إلى الخمسينيات والستينيات من القرن العشرين، عندما بدأ الباحثون لأول مرة في إجراء تجارب على معادلات رياضية لإنشاء الصور، وفي عصرنا الحالي نما هذا المجال ليشمل مجموعة واسعة من التقنيات. يُعدُّ استخدام **الفراكتلات** (Fractals) من أقدم وأشهر تقنيات إنشاء الصور، والفراكتل هو شكل أو نمط هندسي مشابه لذاته، مما يعني أنه يبدو متشابهًا عند تكبيره بمقاييس مُختلفة، وأشهر فراكتل هو الذي يضم مجموعة **ماندلبروت** (Mandelbrot) الموضَّح في الشكل 4.25.



شكل 4.25: فراكتل ماندلبروت

في أواخر القرن العشرين، بدأ الباحثون في استكشاف أساليب أكثر تقدمًا لتوليد الصور مثل الشبكات العصبية.

يُعدُّ **إنشاء صورة من نصّ** (Text-to-Image Synthesis) من أكثر التقنيات شيوعًا لإنشاء الصور باستخدام الشبكات العصبية، وتتضمن هذه التقنية تدريب شبكة عصبية على توليد صور من أوصاف نصيَّة، فتُدرَّب الشبكة العصبية على مجموعة بيانات من الصور والأوصاف النصيَّة المرتبطة بها. وتتعلّم الشبكة ربط كلمات أو عبارات معيّنة بخصائص معيّنة للصورة مثل: شكل العنصر أو لونه، وبمجرد أن تُدرَّب الشبكة يصبح من الممكن استخدامها في إنشاء صور جديدة بناءً على الأوصاف الواردة في النص، وتُستخدم هذه التقنية في إنشاء مجموعة واسعة من الصور تتراوح ما بين العناصر البسيطة إلى المشاهد المعقدة.

وهناك تقنية أخرى لتوليد الصورة تتمثّل في **إنشاء صورة من صورة** (Image-to-Image Synthesis)، وتتضمن هذه التقنية تدريب شبكة عصبية على مجموعة بيانات من الصور؛ لتتعلّم التعرّف على الخصائص الفريدة للصورة حتّى تولّد صورًا جديدة مشابهة للصورة الموجودة، ولكن مع وجود اختلافات. في الآونة الأخيرة استكشف الباحثون **إنشاء صورة من صورة بالاسترشاد بنصّ** (Text-Guided Image-to-Image Synthesis)، مما يجمع بين نقاط القوة في طرائق إنشاء صورة من نصّ، وطرائق إنشاء صورة من صورة من خلال السماح للمستخدِم بتوجيه عملية الإنشاء باستخدام **توجيهات نصيَّة** (Text Prompts)، وتُستخدم هذه التقنية في توليد صور عالية الجودة تتوافق مع التوجيه النصيّ، وتكون في الوقت ذاته مشابهة بصريًا للصورة الطبيعية.

وأخيرًا، هناك تقنية أخرى من أحدث التقنيات في هذا المجال تتمثّل في **رسم صورة بالاسترشاد بنصّ** (Text-Guided Image-Inpainting)، ويُركِّز على ملء الأجزاء المفقودة أو التالفة من الصورة بناءً على وصف نصيٍّ معين، ويقدِّم الوصف النصيّ معلومات عن الشكل الذي يجب أن تبدو عليه الأجزاء المفقودة أو التالفة من الصورة، والهدف من خوارزمية الرسم هذه أن تُستخدم المعلومات؛ لإنشاء صورة واقعية ومترابطة. يقدِّم هذا الدرس أمثلة عملية على توليد الصور من خلال: إنشاء صورة من نصّ، وإنشاء صورة من صورة بالاسترشاد بنصّ، ورسم صور بالاسترشاد بنصّ.

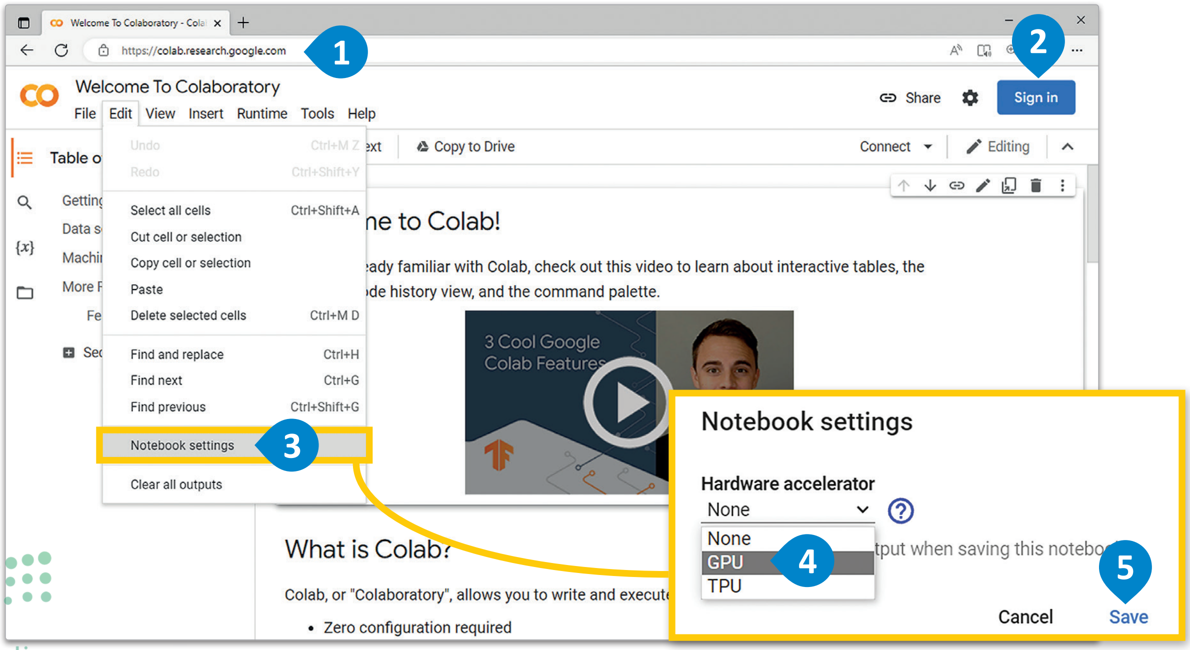
## توليد الصور والموارد الحاسوبية
## Image Generation and Computational Resources

> **وحدة معالجة الرسومات (Graphics Processing Unit - GPU):**
> هي نوع خاص من أنواع المعالِجات مصمَّم للتعامل مع كميات كبيرة من العمليات الحسابية المطلوبة لمعالجة الصور والفيديوهات.

إنشاء الصور مُهمَّة مكلّفة من الناحية الحاسوبيّة؛ لأنها تتضمن استخدام خوارزميات معقَدة تتطلب قدرات عالية من قوة المعالجة، وعادةً تتضمن هذه الخوارزميات معالجة كميات كبيرة من البيانات مثل: نماذج ثلاثية الأبعاد، والنقوش، ومعلومات الإضاءة، مما يمكن أن يؤدي أيضًا إلى زيادة المتطلبات الحاسوبية للمُهِمَّة. يُعدُّ استخدام **وحدات معالجة الرسومات** (Graphics Processing Units - GPUs) أحد التقنيات الرئيسة التي تُستخدم لتسريع توليد الصور. وعلى عكس **وحدة المعالجة المركزية** (Central Processing Unit - CPU) التقليدية المُصمَّمة للتعامل مع مجموعة واسعة من المهام، تم تحسين وحدة معالجة الرسومات حتّى تتناسب مع أنواع العمليات الحسابية المطلوبة لمعالجة الصور والمهام الأخرى المتعلقة بالرسومات، مما يجعلها أكثر كفاءة في التعامل مع كميات كبيرة من البيانات وإجراء عمليات حسابية معقدة، ويُعدُّ هذا سببًا في استخدامها عادة في توليد الصور والمهام الأخرى المكلّفة حاسوبيًّا. يوضِّح هذا الدرس كيف يمكنك استخدام منصة **قوقل كولاب** (Google Colab) الشهيرة للوصول إلى بُنية تحتية قوية قائمة على وحدة معالجة الرسومات دون أي تكلفة، وذلك باستخدام حساب عادي على قوقل، وقوقل كولاب هو منصة مجانية تعتمد على التقنية السحابية، وتتيح للمستخدِمين كتابة المقاطع البرمجية، وتنفيذها، وإجراء التجارب، وتدريب النماذج في بيئة **مفكرة جوبيتر** (Jupyter Notebook).

**للوصول إلى منصة قوقل كولاب:**

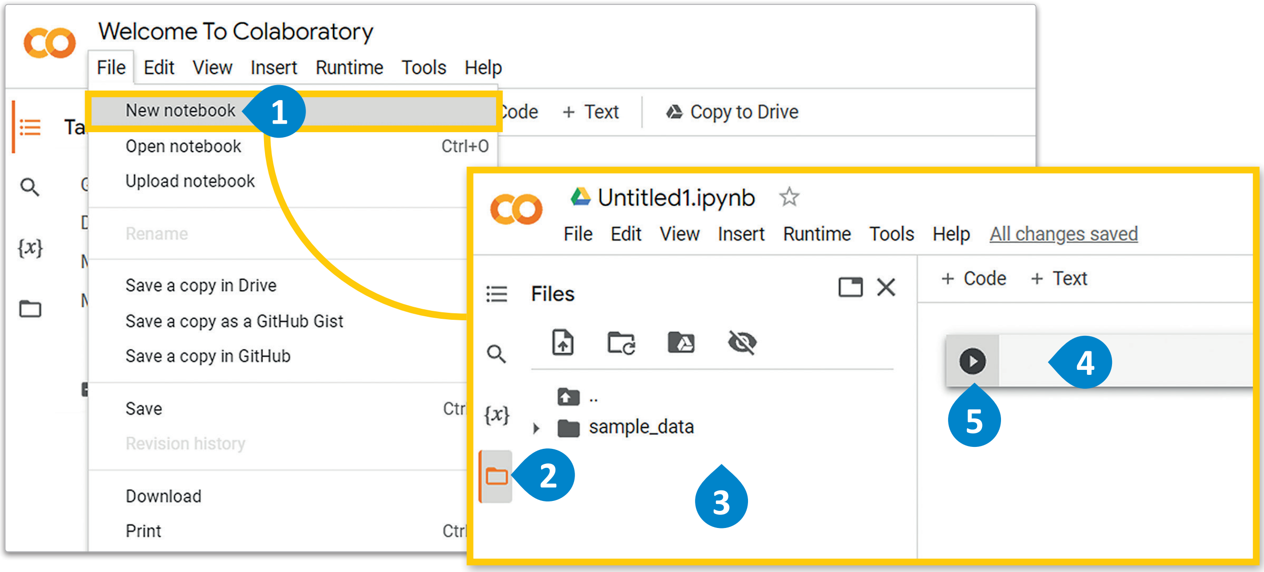
- اذهب إلى: https://colab.research.google.com. 1
- سجِّل الدخول بحساب Google (قوقل) الخاص بك. 2
- اضغط على Edit (تحرير)، ثم Notebook settings (إعدادات المفكرة). 3
- اختر GPU (وحدة معالجة الرسومات)، 4 ثم اضغط على Save (حفظ). 5



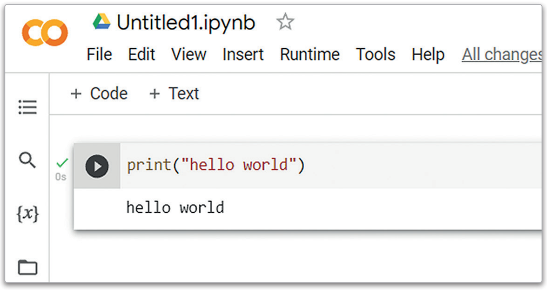

شكل 4.26: الوصول إلى منصة قوقل كولاب

**لاستخدام مفكرة البايثون:**

- اضغط على File (ملف)، ثم على New notebook (مفكرة جديدة). 1
- اضغط على Files (ملفات)، 2 وفي المنطقة المجاورة التي ستظهر لك اسحب وأفلت images (الصور) التي ستستخدمها في الدرس. 3
- يمكنك الآن كتابة مقطعك البرمجي بلغة البايثون داخل خلية المقطع البرمجي، 4 ثم شغّله من خلال الضغط على الزر الموجود بجانب خلية المقطع البرمجي. 5




تعمل بيئة قوقل كولاب بشكل مشابه لعمل مفكرة جوبيتر، وفيما يلي تجد مثال Hello World (مرحبًا بالعالَم) التقليدي:




شكل 4.27: استخدام مفكرة البايثون

**خوارزميات توليد الصور (Image Generation) التي وصفناها في هذا الفصل مصممة بطريقة تجعلها إبداعيّة وبالتالي فهي ليست ثابتة، مما يعني أنه من غير المضمون أن تقوم دائما بتوليد الصورة نفسها للمُدخَلات نفسها. وعليه، فإن الصور المولَّدة المدرجة في هذا الفصل مجرد أمثلة على الصور التي يمكن توليدها باستخدام المقطع البرمجي.**

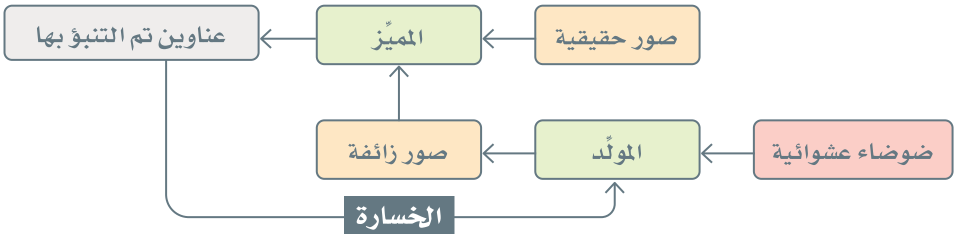
## نماذج الانتشار والشبكة التوليدية التنافسية
## Diffusion Models and Generative Adversarial Networks

في السنوات الأخيرة شهد مجال توليد الصور تقدمًا كبيرًا مع تطوير أساليب ونماذج مُختلفة يمكنها توليد صور واقعية وعالية الجودة من مصادر مُختلفة للمعلومات، وهناك تقنيتان من أكثر التقنيات شيوعًا واستخدامًا على نطاق واسع لتوليد الصور هما: **الشبكة التوليدية التنافسية** (GANs)، ونموذج **الانتشار المستقر** (Stable Diffusion). ستتعرف في هذا القسم على المفاهيم والأساليب الرئيسة الخاصة بالشبكة التوليدية التنافسية ونموذج الانتشار المستقر، كما سيتم تقديم نظرة عامة على تطبيقاتها في توليد الصور، وسيتم مناقشة أوجه التشابه والاختلاف بينهما، ومزايا كل تقنية وعيوبها.

## توليد الصور بالشبكة التوليدية التنافسية
## Generating Images with Generative Adversarial Networks (GANs)

الشبكة التوليدية التنافسية هي فئة من النماذج التوليدية التي تتكون من مكونين رئيسين وهما: **المولِّد** (Generator) **والمميِّز** (Discriminator)، حيث يقوم المولِّد بتوليد صور زائفة، بينما يحاول المميِّز تمييز الصور المولَّدة من الصور الحقيقية، ويُدرَّب هذان المكوِّنان تدريبًا تنافسيًّا، إذ يحاول المولِّد أن "يخدع" المميِّز، ويحاول المميِّز أن يصبح أفضل في اكتشاف الصور الزائفة. تتمثّل إحدى المزايا الرئيسة للشبكة التوليدية التنافسية في قدرتها على توليد صور عالية الجودة وواقعية يصعب تمييزها عن الصور الحقيقية، ولكن يوجد بها أيضًا بعض القيود مثل: **عدم التقارب** (Non-convergence) أو بعبارة أخرى، فشل شبكتي المولِّد والمميِّز في التحسن مع مرور الوقت، و**نقص التنوع** (Mode Collapse) في المُخرَجات، حيث ينتج النموذج نفس المُخرَجات المتشابهة مرارًا وتكرارًا بغض النظر عن المُدخَلات.

**يُطبَّق المولِّد والمميِّز في الشبكة التوليدية التنافسية في العادة باستخدام الشبكات العصبية الترشيحية (CNNs) أو أي معمارية مشابهة.**




شكل 4.28: معمارية الشبكة التوليدية التنافسية

## توليد الصور بالانتشار المستقر Generating Images with Stable Diffusion

الانتشار المستقر هو نموذج تعلُّم عميق لتوليد صورة من نصّ، وتتكون هذه الطريقة من مكونين رئيسين: **مُرمِّز النصّ** (Text Encoder)، **ومفكّك الترميز المرئي** (Visual Decoder). ويُدرَّب مُرمِّز النصّ ومفكِّك الترميز المرئي معًا على مجموعة بيانات مكونة من بيانات نصوص وبيانات صور مقترنة ببعضها؛ حيث يقترن كل مُدخَل نصّي بصورة مقابلة أو أكثر. مُرمِّز النصّ هو شبكة عصبية تأخذ مُدخَلات نصيَّة مثل: جملة أو فقرة وتحوِّلها إلى **تضمين** (Embedding)، والتضمين هو متَّجَه عددي له عدد ثابت من القيم، ويلتقط تمثيل التضمين هذا معنى النصّ المُدخَل. يتم استخدام نهج مشابه في نموذج **الكلمة إلى المتَّجَه** (Word2Vec) ونموذج **ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT) اللذين تم توضيحهما في الوحدة الثالثة، حيث يولِّدان تضمينات للكلمات والجمل الفردية على الترتيب. ويُمرَّر بعد ذلك **تضمين النصّ** (Text Embedding) الذي أنشأه المُرمِّز عبر مفكِّك الترميز المرئي لتوليد صورة، ومفكِّك الترميز المرئي هو أيضًا نوع من الشبكات العصبية ويُنفذ عادةً باستخدام **شبكة عصبية ترشيحية** (CNN) أو معمارية مشابهة، وتُقارن الصورة المولَّدة بالصورة الحقيقية المقابلة الموجودة في مجموعة البيانات، ويُستخدم الفرق بينهما لحساب **الخسارة** (Loss)، ثم تُستخدم الخسارة لتحديث متغيِّرات مُرمِّز النصّ ومفكِّك الترميز المرئي؛ لتقليل الاختلاف بين الصور التي وُلِّدت والصور الحقيقية.

**جدول 4.4: عملية تدريب الانتشار المستقر**

| |
|---|
| 1. مرِّر المُدخَلات النصيَّة عبر مُرمِّز النصّ للحصول على تضمين النصّ. |
| 2. مرِّر تضمين النصّ عبر مفكِّك الترميز المرئي لتوليد صورة. |
| 3. احسب الخسارة (الاختلاف) بين الصورة المولَّدة والصورة الحقيقية المقابلة لها الموجودة في مجموعة البيانات. |
| 4. استخدِم الخسارة؛ لتحديث متغيِّرات مُرمِّز النصّ ومفكِّك الترميز المرئي، وعندما يكون المستوى عاليًا يتضمن ذلك **مكافأة** (Rewarding) الخلايا العصبية التي ساعدت على تقليل الخسارة و**معاقبة** (Punishing) الخلايا العصبية التي ساهمت في زيادتها. |
| 5. كرِّر الخطوات المذكورة سابقًا مع أزواج متعددة من النصوص والصور في مجموعة البيانات. |

حقَّق كلٌّ من نموذج الشبكة التوليدية التنافسية ونموذج الانتشار المستقر نتائج مبهرة في مجال توليد الصور، ويُركِّز الجزء المتبقي من هذا الدرس على تقديم أمثلة عملية بلغة البايثون على النهج **القائم على الانتشار** (Diffusion-Based) والذي يُعدُّ حاليًا أحدث ما توصلت إليه التقنية. كما تم التوضيح من قبل، يُعدُّ توليد الصور مُهمَّة مكلِّفة حاسوبيًّا، ولذلك نوصيك بشدة بأن تطبق جميع أمثلة البايثون على نظام قوقل كولاب الأساسي أو أي بنية أساسية مُختلفة تدعمها وحدة معالجة رسومات يكون لديك حق الوصول إليها.

يستخدِم هذا الفصل مكتبة diffusers التي تُعدُّ حاليًا أفضل مكتبة مفتوحة المصدر للنماذج القائمة على الانتشار، ويقوم المقطع البرمجي التالي بتثبيت المكتبة، وكذلك بعض المكتبات الإضافية المطلوبة:

```
%%capture
!pip install diffusers
!pip install transformers
!pip install accelerate

import matplotlib.pyplot as plt
from PIL import Image # used to represent images
```

## توليد الصورة من نصّ Text-to-Image Generation

يوضِّح هذا القسم الطريقة التي يمكن بها استخدام مكتبة diffusers لتوليد صور تعتمد على التوجيه النصيّ الذي يقدمه المستخدِم، وتُستخدم الأمثلة الواردة في هذا القسم نموذج stable-diffusion-v1-4 (الانتشار-المستقر- الإصدار 1-4)، وهو نموذج شائع مُدرَّب مسبقًا لتوليد الصورة من نصّ.

```
# a tool used to generate images using stable diffusion
from diffusers import DiffusionPipeline
generator = DiffusionPipeline.from_pretrained("CompVis/stable-diffusion-v1-4")
# specifies what GPUs should be used for this generation
generator.to("cuda")

image = generator("A photo of a white lion in the jungle.").images[0]
plt.imshow(image);
```

يستجيب النموذج للتوجيه A photo of a white lion in the jungle (صورة أسد أبيض في الغابة) بصورة مبهرة وواقعية جدًا، كما هو موضَّح في الشكل 4.29، ويُعدُّ التجريب باستخدام التوجيهات الإبداعية هو أفضل طريقة لاكتساب الخبرة وفهم قدرات هذا النهج ونقاط ضعفه.

شكل 4.29: صورة مولّدة لأسد أبيض في الغابة

**معلومة**

معمارية أجهزة الحاسب الموحد (Compute Unified Device Architecture - CUDA) هي منصة حوسبة موازية تتيح استخدام وحدات معالجة الرسومات (GPUs).



وزارة التعليم
Ministry of Education
2025 - 1447

يضيف **التوجيه** (Prompt) التالي بُعدًا إضافيًّا لعملية التوليد، إذ يطلب أن يُرسم أسد أبيض بطريقة **بابلو بيكاسو** (Pablo Picasso)، وهو من أشهر الرسامين في القرن العشرين.

```
image = generator("A painting of a white lion in the style of Picasso.").
images[0]
plt.imshow(image);
```

ومرة أخرى، النتائج مبهرة وتُظهر الإبداع في عملية الانتشار المستقر، فالصورة الناتجة عن العملية هي في الواقع صورة أسد أبيض. ولكن على عكس التوجيه السابق، يؤدي التوجيه الجديد إلى صور تشبه الرسم بدلًا من أن تشبه الصور الفوتوغرافية، بالإضافة إلى ذلك، فإن أسلوب اللوحة يشبه بالفعل وبشكل ملحوظ أسلوب بابلو بيكاسو.

شكل 4.30: صورة مولَّدة لأسد على نمط بيكاسو

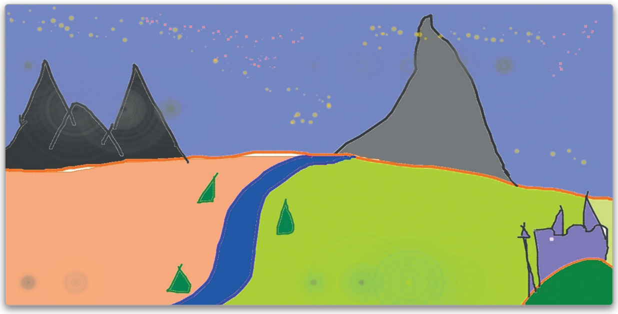
## توليد صورة من صورة من خلال الاسترشاد بنصّ
## Image-to-Image Generation with Text Guidance

يستخدِم المثال التالي مكتبة diffusers لتوليد صورة بناءً على مُدخَلين هما: صورة موجودة تعمل كأساس للصورة الجديدة التي سيتم إنشاؤها، وتوجيه نصيّ يصف الشكل الذي يجب أن تبدو عليه الصورة المنتَجة. بما أن مُهِمَّة تحويل النصّ إلى الصورة الموضَّحة في القسم السابق كانت محدودة فقط بتوجيه نصيّ، فيجب أن تضمن المُهِمَّة الجديدة أن تكون الصورة الجديدة مشابهة للصورة الأصلية، ومُمثِّلةً بشكلٍ دقيق للوصف الوارد في التوجيه النصيّ.

```
# pipeline used for image to image generation with stable diffusion
from diffusers import StableDiffusionImg2ImgPipeline
# loads a pretrained generator model
generator = StableDiffusionImg2ImgPipeline.from_pretrained("runwayml/stable-
diffusion-v1-5")
# moves the generator model to the GPU (CUDA) for faster processing
generator.to("cuda")

init_image = Image.open("landscape.jpg")
init_image.thumbnail((768, 768)) # resizes the image to prepare it as input of the model
plt.imshow(init_image);
```


وزارة التعليم
Ministry of Education
2025 - 1447

المثال الموجود في الشكل 4.31 يستخدِم النموذج المدرَّب مسبقًا stable-diffusion-v1-4 المناسب لتوليد صورة من صورة من خلال التوجيه النصي.



شكل 4.31: صورة المنظر الطبيعي الأصلية

```
# a detailed prompt describing the desired visual
# for the produced image
prompt = "A realistic mountain
landscape with a large castle."
image = generator(prompt=prompt,
image = init_image, strength=0.75).
images[0]
plt.imshow(image);
```

شكل 4.32: صورة منظر طبيعي مولَّدة بقوة = 0.75

في الواقع، يولِّد النموذج صورة مستجيبةً للتوجيه النصيّ ومشابهة بصريًا للصورة الأصلية، ويُستخدم متغيِّر strength (القوة) للتحكم في الاختلاف البصري بين الصورة الأصلية والصورة الجديدة، ويتخذ المتغيِّر قيمًا بين 0 و1، وتسمح القيم الأعلى للنموذج بأن يكون أكثر مرونة وأقل تقيُّدًا بالصورة الأصلية. على سبيل المثال، يُستخدم المقطع البرمجي التالي لنفس prompt (التوجيه) من خلال ضبط المتغيِّر strength ليساوي 1.

```
# generate a new image based on the prompt and the
# initial image using the generator model
image = generator(prompt=prompt,
image = init_image, strength=1).images[0]
plt.imshow(image);
```

شكل 4.33: صورة منظر طبيعي مولَّدة بقوة = 1

تؤكد الصورة الناتجة في شكل 4.33 أن زيادة قيمة متغيِّر القوة تؤدي إلى شكل بصري أفضل بالإرشاد الوارد في التوجيه النصي، ولكنه أيضًا أقل تشابهًا إلى حد كبير مع الصورة المُدخَلة.

وهذا مثال نموذجي آخر، يتضح مُخرَجه في الشكل 4.34.

```
init_image = Image.open("cat_1.jpg")
init_image.thumbnail((768, 768))
plt.imshow(init_image);
```

شكل 4.34: صورة القطّة الأصلية



وزارة التعليم
Ministry of Education
2025 - 1447

وسيُستخدم المقطع البرمجي التالي لتحويل هذه الصورة إلى صورة tiger (نمر):

```
prompt = "A photo of a tiger"
image = generator(prompt=prompt, image=init_image, strength=0.5).images[0]
plt.imshow(image);
```

تتقيد المحاولة الأولى بقيمة المتغيِّر strength، مما أدّى إلى صورة تبدو وكأنها مزيج بين النمر والقطّة الموجودة في الصورة الأصلية، كما هو موضَّح في الشكل 4.35، وتَدُل الصورة الجديدة على أن الخوارزمية لم تكن لديها القوة الكافية لتحويل وجه القطّة تحويلًا صحيحًا إلى وجه نمر، وتظل الخلفية مشابهة جدًا لخلفية الصورة الأصلية.

شكل 4.35: صورة نمر مولَّدة بقوة = 0.5

بعد ذلك، تتم زيادة المتغيِّر strength للسماح للنموذج بالابتعاد عن الصورة الأصلية والاقتراب أكثر من التوجيه النصيّ.

```
image = generator(prompt=prompt,
image = init_image, strength=0.75).
images[0]
plt.imshow(image);
```

في الواقع، الصورة الجديدة المعروضة هي صورة نمر، ولكن لاحظ أن البيئة المحيطة بالحيوان ووضعية جلوسه وزواياه تظل شديدة الشبه بالصورة الأصلية، ويَدُل ذلك على أن النموذج ما زال واعيًا بالصورة الأصلية وحاولَ أن يحافظ على عناصر كان لا بد ألا تُغير؛ حتّى يقترب أكثر من التوجيه النصيّ.

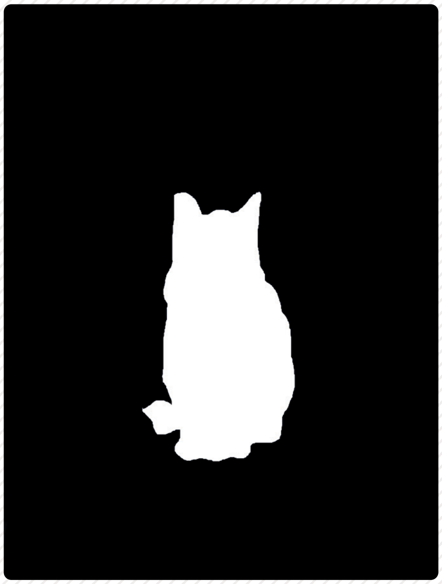
شكل 4.36: صورة النمر مولَّدة بقوة = 0.75

## رسم صورة بالاسترشاد بنصّ Text-Guided Image-Inpainting

يُركِّز المثال التالي على استخدام نموذج الانتشار المستقر لاستبدال شكل بصري جديد يصفه التوجيه النصيّ بأجزاء محددة من صورة معيّنة، ويُستخدم لهذا الغرض النموذج المدرَّب مسبقًا stable-diffusion-inpainting (رسم-الانتشار- المستقر)، ويقوم المقطع البرمجي التالي بتحميل صورة قطّة على مقعد، وهناك **قناع** (Mask) يعزل الأجزاء المحددة من الصورة التي تغطيها القطّة:

```
# tool used for text-guided image in-painting
from diffusers import StableDiffusionInpaintPipeline
init_image = Image.open("cat_on_bench.png").resize((512, 512))
plt.imshow(init_image);
mask_image = Image.open("cat_mask.jpg").resize((512, 512))
plt.imshow(mask_image);
```

شكل 4.37: صورة القطّة الأصلية


شكل 4.38: قناع صورة القطّة

**القناع** (Mask) هو صورة بسيطة بالأبيض والأسود لها نفس أبعاد الصورة الأصلية بالضبط، والأجزاء التي استُبدلت في الصورة الجديدة تُميز باللون الأبيض، في حين أن الأجزاء الأخرى من القناع سوداء. بعد ذلك، يتم تحميل النموذج المدرَّب مسبقًا، ويتم إنشاء prompt (التوجيه) لكي توضع صورة رائد الفضاء مكان القطة التي في الصورة الأصلية، كما يظهر في الشكل 4.39.

```
generator = StableDiffusionInpaintPipeline.from_pretrained("runwayml/stable-diffusion-inpainting")
generator = generator.to("cuda")

prompt = "A photo of an astronaut"
image = generator(prompt=prompt, image=init_image, mask_image=mask_image).images[0]
plt.imshow(image);
```

وزارة التعليم
Ministry of Education
2025 - 1447

نجحت الصورة الجديدة في أن تظهر صورة واقعية للغاية لرائد الفضاء الذي وضعته مكان القطة التي كانت في الصورة الأصلية، كما يمتزج هذا الشكل البصري بسلاسة مع عناصر الخلفية والإضاءة في الصورة.

في الواقع، حتّى لو كان القناع أبسط وأقل دقة، يمكن إنتاج بديل واقعي. لاحظ صورة المُدخَل والقناع التاليين:

شكل 4.39: صورة رائد فضاء مولَّدة

```
init_image = Image.open("desk.jpg").resize((512, 512))
plt.imshow(init_image);
mask_image = Image.open("desk_mask.jpg").resize((512, 512))
plt.imshow(mask_image);
```

شكل 4.40: صورة المكتب الأصلية

شكل 4.41: قناع صورة المكتب

في هذا المثال، يغطي القناع جهاز الحاسب المحمول الموجود في وسط الصورة، ثم يُستخدم prompt (التوجيه) التالي والمقطع البرمجي ليتم وضع صورة الكتاب مكان جهاز الحاسب المحمول الموجود في الصورة الأصلية:

```
prompt = "A photo of a book"
image = generator(prompt=prompt, image=init_image, mask_image=mask_image).
images[0]
plt.imshow(image);
```

على الرغم من أن prompt (التوجيه) طلب إدخال كائن (كتاب) يختلف اختلافًا كبيرًا عن الكائن الذي استُبدل وهو (جهاز الحاسب المحمول)، فقد قام النموذج بعمل جيد في مزج الأشكال والألوان؛ لإنشاء شكل بصري دقيق، ومع التقُدم المستمر في تقنيات تعلُّم الآلة ورسومات الحاسب، من المحتمل أن تُنشِئَ صورًا أكثر إبهارا وأكثر واقعية في المستقبل.

شكل 4.42: صورة مكتب مولَّدة وعليها كتاب

# تمرينات

1 صِف باختصار عملية رسم صورة بالاسترشاد بنصّ.

2 صِف عملية تدريب نماذج الانتشار المستقر.

3 صِف المولِّد والمميِّز في الشبكة التوليدية التنافسية.

4 استخدِم أداة DiffusionPipeline من مكتبة diffusers لإنشاء صورة لحيوانك المفضل وهو يأكل طعامك المفضل. يمكنك استخدام منصة قوقل كولاب في هذه المُهِمَّة.

5 استخدِم أداة StableDiffusion2ImagePipleline من مكتبة diffusers لتحويل الحيوان في الصورة المرسومة في التمرين السابق إلى حيوان آخر من اختيارك. يمكنك استخدام منصة قوقل كولاب في هذه المُهِمَّة.



وزارة التعليم
Ministry of Education
2025 - 1447

# المشروع

لا تستجيب كل مجموعة بيانات بالطريقة نفسها للتدريب بكل خوارزميات التصنيف، ولكي تحصل على أفضل النتائج لمجموعة بياناتك عليك أن تجرِّب استخدام خوارزميات مُختلفة، وتُقدم لك مكتبة Sklearn في البايثون مجموعة متنوّعة من الخوارزميات التي يمكنك تجريبها، بما فيها الخوارزميات التالية:

> من sklearn.ensemble.forest استورد خوارزمية RandomForestClassifier.

> من sklearn.naive_bayes استورد خوارزمية GaussianNB.

>من sklearn.svm استورد خوارزمية SVC.

1 استخدِم مجموعة تدريب وجوه الحيوانات لتدريب نموذج يحقق أكبر دقة ممكنة على مجموعة الاختبار.

2 استبدِل خوارزمية SGDClassifier بكل من الخوارزميات المذكورة أعلاه (RandomForestClassifier، GaussianNB، SVC) وحاول أن تحدِّد أفضلها.

3 أعِد تشغيل مفكرتك بعد كل عملية استبدال لحساب دقة كل نموذج جديد تجرِّبه.

4 أنشئ تقريرًا يقارن دقة كل النماذج التي جرّبتَها وحدِّد النموذج الذي حقق أفضل دقة.

وزارة التعليم
Ministry of Education
2025 - 1447

# ماذا تعلّمت

- إعداد الصور للتعرُّف عليها.
- استخدام المكتبات والدوال لإنشاء نماذج التعلُّم الموجَّه لتصنيف الصور.
- وصف طريقة تركيب الشبكات العصبية.
- استخدام المكتبات والدوال لإنشاء نماذج التعلُّم غير الموجَّه لعنقدة الصور.
- إنشاء الصور من خلال توفير التوجيه النصيّ.
- إكمال الأجزاء الناقصة لصورة ببيانات واقعية.

## المصطلحات الرئيسة

| | |
|---|---|
| صورة | Image |
| توليد الصور | Image Generation |
| المعالجة الأوليَّة للصور | Image Preprocessing |
| طبقة الشبكة | Network Layer |
| التعرّف | Recognition |
| الانتشار المستقر | Stable Diffusion |
| تحجيم قياسي | Standard Scaling |
| بيانات مرئية | Visual Data |

| | |
|---|---|
| رؤية الحاسب | Computer Vision |
| الشبكة العصبية الترشيحية | Convolutional Neural Network - CNN |
| نموذج الانتشار | Diffusion Model |
| هندسة الخصائص | Feature Engineering |
| انتقاء الخصائص | Feature Selection |
| الشبكة التوليدية التنافسية | Generative Adversarial Network - GAN |
| مُخطَّط تكراري للتدرجات الموجَّهة | Histogram of Oriented Gradients - HOG |



وزارة التعليم
Ministry of Education
2025 - 1447

# 5. خوارزميات التحسين واتخاذ القرار

سيتعرّف الطالب في هذه الوحدة على عدة خوارزميات وتقنيات تساعده في إيجاد أكثر الحلول كفاءة لمشكلات التحسين المعقدة، كما سيتعلّم طريقة عمل خوارزميات التحسين، وخوارزميات اتخاذ القرار، وطريقة تطبيقها لحلّ مشكلات متعلقة بالعالم الواقعي ترتبط بتخصيص الموارد والجدولة وتحسين المسارات.

## أهداف التعلُّم

بنهاية هذه الوحدة سيكون الطالب قادرًا على أن:

- يُصنِّف طرائق التحسين لمعالجة مشكلات معقدة.
- يَصِف خوارزميات اتخاذ القرار المُختلفة.
- يَستخدِم البايثون لحلّ مشكلات تخصيص الموارد المتعلقة بفِرق العمل.
- يَحلّ مشكلات الجدولة باستخدام خوارزميات التحسين.
- يَستخدِم البايثون لحلّ مشكلات الجدولة.
- يَستخدِم البرمجة الرياضية لحلّ مشكلات التحسين.
- يُعرِّف مشكلة حقيبة الظهر (Knapsack Problem).
- يُعرِّف مشكلة البائع المُتجوّل (Traveling Salesman Problem).

## الأدوات

- مفكّرة جوبيتر (Jupyter Notebook)

وزارة التعليم
Ministry of Education
2025 - 1447

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الأول

# مشكلة تخصيص الموارد

## خوارزميات التحسين في الذكاء الاصطناعي
## Optimization Algorithms in AI

يُستخدم الذكاء الاصطناعي في مُختلف الصناعات لاتخاذ قرارات تتسم بالكفاءة والدقة، ويُعدُّ استخدام خوارزميات تعلُّم الآلة إحدى طرائق الذكاء الاصطناعي المُستخدَمة في اتخاذ القرارات. وكما تعلّمت في الوحدة السابقة، فإن خوارزميات تعلُّم الآلة تقوم بتمكين الذكاء الاصطناعي من التعلُّم بواسطة البيانات ومن ثَمّ القيام بالتنبؤات أو تقديم التوصيات. على سبيل المثال، في مجال الرعاية الصحية، يُمكن استخدام الذكاء الاصطناعي للتنبؤ بنتائج المرضى والتوصية بخُطط علاجية بناءً على البيانات التي جُمعت من حالات مماثلة. وفي مجال التمويل، يُمكن استخدام الذكاء الاصطناعي في اتخاذ قرارات استثمارية بواسطة تحليل مجموعات كبيرة من البيانات المالية وتحديد الأنماط التي تبيّن المخاطر أو الفرص المحتملة. وعلى الرغم من أن خوارزميات تعلُّم الآلة تحظى بشعبية متزايدة إلا أنها ليست النوع الوحيد من خوارزميات الذكاء الاصطناعي التي يُمكن استخدامها في اتخاذ القرارات، فهناك طريقة أخرى تتمثل في استخدام خوارزميات التحسين التي تُستعمَل بوجه عام لإيجاد أفضل حلّ لمشكلة محدَّدة بناءً على قيود وأهداف معيّنة.

يهدف التحسين إلى تحقيق التصميم الأفضل بالنسبة لمجموعة من المعايير أو القيود ذات الأولوية، وتشمل تعزيز عوامل معيّنة مثل: الإنتاجية، والموثوقية، وطول العمر، والكفاءة، وفي الوقت نفسه تقليل عوامل أخرى مثل: التكاليف، والفاقد، والتوقف عن العمل، والأخطاء.

**القيود (Constraints):**
هي بمثابة شروط تقيِّد الحل، مثل الحد الأقصى لوزن الطرد الذي يمكن شحنه.

**الدوال الموضوعية (Objective Functions):**
هي معايير تحدِّد مدى اقتراب الحل المقدَّم من النتائج المطلوبة، مثل تقليل مسافة السفر لشاحنة توصيل.

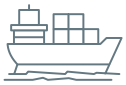
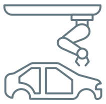
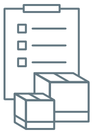
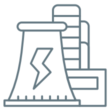
### مشكلات التخصيص Allocation Problems

تُعدُّ مشكلات التخصيص من مشكلات التحسين الشائعة؛ ففيها يتم تخصيص مجموعة من الموارد مثل: العمّال، أو الآلات، أو الأموال لمجموعة من المهام أو المشاريع بأعلى كفاءة ممكنة، وتنشأ هذه المشكلات في مجموعة واسعة من المجالات بما فيها التصنيع والخدمات اللوجستية وإدارة المشاريع والتمويل، ويُمكن صياغتها بطرائق مُختلفة بناءً على قيودها وأهدافها. في هذا الدرس ستتعرّف على مشكلات التخصيص وخوارزميات التحسين المستخدَمة لحلّها.

شكل 5.1: استخدام خوارزميات التحسين في مستودع

وزارة التعليم
Ministry of Education
2025 - 1447

بعد ذلك، ستشاهد عددًا من الأمثلة، ولكل مثال منها قيود ودوال موضوعية خاصة به.

| | القيود | الدوال الموضوعية |
|---|---|---|
|  شركات النقل | - وضع أُطر زمنية للتوصيل؛ لضمان توصيل الطرود وفق إطار زمني محدَّد.<br>- توفير سِعة مركبات التوصيل؛ لضمان استخدام المركبة المناسبة لكل عملية توصيل، ومقدرتها على حمل الكمية اللازمة من الطرود.<br>- توفير السائقين والموظفين، ومراعاة تقسيم أوقات عملهم؛ لضمان كفاءة العمل، وعدم تكليفهم بأعمال فوق قدرتهم. | - **تقليل** (Minimizing) وقت التوصيل ومسافة السفر؛ لخفض التكلفة وتحسين الكفاءة.<br>- **زيادة** (Maximizing) عدد الطرود في كل مركبة؛ لتقليل عدد الرحلات اللازمة.<br>- **زيادة** (Maximizing) رضا العملاء من خلال توصيل الطرود في وقت محدَّد وفق إطار زمني محدَّد. |
| جدولة خطوط الطيران | - تَوَفُّر الطائرات وجداول الصيانة؛ لضمان إجراء الصيانة الجيدة لها، ومدى جاهزيّتها للرحلات.<br>- قيود مراقبة الحركة الجوية؛ لتجنُّب التأخير وتقليل استهلاك الوقود.<br>- مراعاة حاجة المسافر وتفضيلاته؛ لجدولة رحلات الطيران الأنسب للمسافرين. | - **تقليل** (Minimizing) تأخر رحلات الطيران أو إلغائها؛ لزيادة رضا العملاء.<br>- **زيادة** (Maximizing) استغلال الطائرات؛ لتقليل التكاليف وتحسين الكفاءة.<br>- **زيادة** (Maximizing) الإيرادات من خلال عمل عروض خاصة على رحلات الطيران عالية الطلب، وتعديل أسعار التذاكر بناءً على الطلب. |
|  المصنِّعون | - سعة الإنتاج والمهلة الزمنية؛ لضمان تصنيع المُنتَجات في الوقت المناسب.<br>- توفير المواد وسعة التخزين؛ لتجنُّب نفاد المخزون أو تكدسه.<br>- تقلّبات الطلب؛ لتعديل جداول الإنتاج بناء على التغيرات في طلبات العملاء. | - **تقليل** (Minimizing) تكاليف الإنتاج من خلال تحسين استخدام الموارد وتقليل الفاقد.<br>- **زيادة** (Maximizing) كفاءة الإنتاج من خلال جدولة دورات الإنتاج؛ لتقليل أوقات التجهيز والتبديل.<br>- **زيادة** (Maximizing) رضا العملاء من خلال ضمان توفير المُنتَجات عند الحاجة إليها. |
|  إدارة المخزون في الشركات | - سعة تخزين محدودة تتطلب إدارة دقيقة لمستويات المخزون.<br>- فترات مهلة التسليم وتنوّعها، التي تؤثر على مقدار المخزون الذي يجب الاحتفاظ به في أي وقت.<br>- توفير ميزانية؛ لشراء مخزون. | - **زيادة** (Maximizing) الربح من خلال ضمان وجود مستويات كافية من مخزون السلع ذات هامش الربح العالي.<br>- **تقليل** (Minimizing) تكاليف التخزين من خلال تحسين مستويات المخزون بناءً على توقُّعات الطلب.<br>- **زيادة** (Maximizing) رضا العملاء من خلال ضمان توفُّر المُنتَجات المناسبة في الوقت المناسب وفي المكان المناسب، وبتقليل نفاد المخزون والتأخير والمشكلات الأخرى التي قد تؤثر على تجربة العملاء. |
|  شركات الطاقة | - مراعاة الطلب على الكهرباء وتقلّباته.<br>- توفُّر المواد الخام وموارد الطاقة الضرورية.<br>- قيود النقل والتوزيع مثل: سعة الشبكة والمسافة بين مصانع توليد الطاقة والمستهلكين. | - **تقليل** (Minimizing) تكلفة توليد الكهرباء وتوزيعها من خلال تحسين استخدام الموارد.<br>- **تقليل** (Minimizing) هدر الطاقة وفشل الخدمات. |

يُمكن نمذجة كل التطبيقات الواردة سابقًا في صورة مشكلات معقدة لها عدد كبير من الحلول المُمكنة. على سبيل المثال، فكِّر في مشكلة تخصيص الموارد المعهودة التي تركّز على تشكيل فريق، حيث تنشأ المشكلة عندما يكون لديك:

- مجموعة كبيرة من العمّال يمتلكون مهارات مُختلفة.
- مُهِمَّة تتطلب مجموعةً فرعية محدَّدة من المهارات لأجل إكمالها.

ويتمثّل الهدف في تكوين فريق بأقل عدد ممكن من العمّال، مع الالتزام في الوقت نفسه **بالقيد** (Constraint) الذي ينصّ على توفّر جميع المهارات المطلوبة في أعضاء الفريق؛ لأداء المُهِمَّة.

على سبيل المثال، تخيل سيناريو بسيطًا يوجد فيه خمسة عمال:

تتطلب المُهِمَّة المراد إنجازها كل المهارات: م1، م2، م3، م4، م5، م6. يتمثّل الحلّ القائم على **القوة المُفرطة** (Brute Force) في أخذ كل فِرق العمّال المُمكنة في الاعتبار، والتركيز على الفِرق التي تتوفّر فيها جميع المهارات المطلوبة، واختيار الفريق الأقل عددًا، وعلى افتراض أن كل فريق يتكون من شخص واحد على الأقل، فيُمكنك أن تُشكِّل واحدًا وثلاثين فريقًا مختلفًا يتكون كل منهم من خمسة عمّال.

**القوة المُفرطة (Brute-force):**

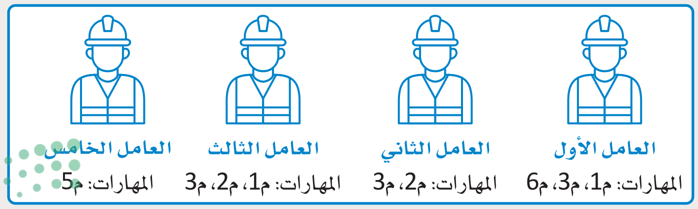
هي طريقة من طرائق حلّ المشكلات تتضمن التجريب المنهجي لجميع الحلول المكنة للمشكلة بهدف الوصول إلى الحلّ الأمثل، بغضّ النظر عن التكلفة الحاسوبية.

- بالنسبة للفريق المُكوَّن من عامل واحد، هناك خمس طرائق لاختيار عامل واحد من بين العمّال الخمسة.
- بالنسبة للفريق المُكوَّن من عاملين اثنين، هناك عشر طرائق لاختيار عاملين من بين العمّال الخمسة.
- بالنسبة للفريق المُكوَّن من ثلاثة عمّال، هناك عشر طرائق لاختيار ثلاثة عمّال من بين العمّال الخمسة.
- بالنسبة للفريق المُكوَّن من أربعة عمّال، هناك خمس طرائق لاختيار أربعة عمّال من بين العمّال الخمسة.
- بالنسبة للفريق المُكوَّن من خمسة عمّال، هناك طريقة واحدة لاختيار كل العمّال الخمسة.

**العدد الإجمالي للفِرق المُختلفة التي يُمكنك تكوينها هو: 5+10+10+5+1=31، ويُمكن حساب العدد أيضًا وفقًا للمعادلة: $2^5 - 1$.**

يكشف تقييم كل الفِرق الإحدى والثلاثين عن أفضل حلّ ممكن يتمثّل في تكوين فريق يشمل العمّال: الأول والرابع والخامس، وسيغطي هذا الفريق كل المهارات الست المطلوبة، وسيشمل الفريق ثلاثة عمّال، ولا يُمكن تغطية كل المهارات بفريق يشتمل على عدد عمّال أقل من ذلك، مما يجعل هذا الحلّ هو **الحلّ الأمثل** (Optimal Solution).

وهناك حلّ آخر يتمثّل في تكوين فريق يشمل العمّال: الأول والثاني والثالث والخامس، وعلى الرغم من أن هذا الفريق يغطي كل المهارات الست، إلا أنه يتطلب أيضًا عمّالًا أكثر، مما يجعل هذا الحلّ ممكنًا، ولكنه ليس الحلّ الأمثل.



الطبيعة الخاصّة بأسلوب القوة المُفرطة تضمن دائمًا إيجاد الحلّ الأمثل، متى أمكن ذلك، ولكنّ فحص كل الفرق المُمكنة يُعدُّ عملية مكلّفة حاسوبيًا، فمثلًا:

- إذا كان لديك ستة عمّال، فسيكون عدد الفرق المُمكنة: $2^6 - 1 = 63$.
- إذا كان لديك عشرة عمّال، فسيكون عدد الفرق المُمكنة: $2^{10} - 1 = 1{,}023$.
- إذا كان لديك خمسة عشر عاملًا، فسيكون عدد الفرق المُمكنة: $2^{15} - 1 = 32{,}767$.
- إذا كان لديك عشرون عاملًا، فسيكون عدد الفرق المُمكنة: $2^{20} - 1 = 1{,}048{,}575$.
- إذا كان لديك خمسون عاملًا، فسيكون عدد الفرق المُمكنة: $2^{50} - 1 = 1{,}125{,}899{,}906{,}842{,}623$.

من الواضح في مثل هذه المواقف أن حصر عدد الفرق لكل الحلول المُمكنة ليس خيارًا عمليًّا، ولذلك تم اقتراح طرائق تحسين أخرى لمعالجة المشكلات المعقدة عن طريق البحث في خيارات الحلول المُمكنة بأسلوب أكثر كفاءة من أسلوب القوة المُفرطة، ويُمكن بوجه عام تصنيف هذه الطرائق في ثلاث فئات:

حتى بالنسبة لعدد معتدل من 50 عاملًا، فإن عدد الفرق المحتملة يتضخم إلى أكثر من كوادريليون ($\text{Quadrillion}=10^{15}$).

- طرائق الاستدلال (Heuristic Methods)
- البرمجة القيدية (Constraint Programming)
- البرمجة الرياضية (Mathematical Programming)

## الحلّ الأمثل Optimal Solution

من الممكن أن تكون هناك العديد من الحلول المُثلى، كأن يكون لديك عدة فرق تشمل ثلاثة عمّال وبإمكانها أن تستوفي كل المهارات المطلوبة، كما أنه من الممكن ألا يوجد حلّ لبعض المشكلات، على سبيل المثال: إذا كانت المُهِمَّة تتطلب المهارة السابعة وهي لا تتوفّر في أي عامل من العمّال، فلن يكون هناك حلّ للمشكلة.

### طرائق الاستدلال (Heuristic Methods)

تقوم **طرائق الاستدلال** (Heuristic Methods – HM) في العادة على التجربة، أو البديهة، أو الفطرة السليمة، وليس على التحليل الرياضي الدقيق، ويُمكن استخدامها لإيجاد حلول جيدة بشكل سريع، ولكنها لا تضمن الوصول إلى الحلّ الأمثل (أفضل حل يمكن الحصول عليه)، ومن الأمثلة على الخوارزميات الاستدلالية: **الخوارزميات الجشعة** (Greedy Algorithms)، و**محاكاة التلدين** (Simulated Annealing)، و**الخوارزميات الجينية** (Genetic Algorithms)، و**تحسين مستعمرة النمل** (Ant Colony Optimization). تُستخدم هذه الطرائق في العادة لحلّ المشكلات المعقدة التي تستغرق وقتًا حاسوبيًا طويلًا جدًّا، ولكن لا يُمكنها إيجاد حلول دقيقة، وستتعلّم في الدروس القادمة المزيد عن هذه الخوارزميات.

**+ الإيجابيات**

تتميز الاستدلالات بالكفاءة الحاسوبية، ويُمكنها أن تتناول المشكلات المعقدة، كما يُمكنها أن تجد حلولًا ذات جودة عالية إذا استُخدمت لها استدلالات معقولة.

**- السلبيات**

لا تضمن الوصول إلى الحلّ الأمثل، كما أن بعض الاستدلالات تتطلب ضبطًا كبيرًا حتى تُؤدي إلى نتائج جيدة.

### البرمجة القيدية (Constraint Programming)

**البرمجة القيدية** (Constraint Programming – CP) تحلّ مشكلات التحسين عن طريق نمذجة القيود وإيجاد حلّ يخضع لجميع القيود، وهذا الأسلوب مفيد بشكل خاص في المشكلات التي بها عدد كبير من القيود أو التي تتطلب تحسين عدة أهداف.

**+ الإيجابيات**

يُمكن للبرمجة القيدية أن تتعامل مع قيود معقدة وأن تجد أفضل الحلول.

**- السلبيات**

يُمكن أن تكون هذه الطرائق مكلّفة حاسوبيًا في المشكلات الكبيرة.



وزارة التعليم
Ministry of Education
2025 - 1447

## البرمجة الرياضية (Mathematical Programming)

**البرمجة الرياضية** (Mathematical Programming – MP) هي مجموعة من التقنيات التي تَستخدم نماذج رياضية؛ لحلّ مشكلات التحسين، وتشمل: **البرمجة الخطية** (Linear Programming)، **والبرمجة الرباعية** (Quadratic Programming)، **والبرمجة غير الخطية** (Nonlinear Programming) **وبرمجة الأعداد الصحيحة المختلطة** (Mixed-Integer Programming)، وتُستخدم هذه التقنيات على نطاق واسع في الكثير من المجالات؛ بما فيها علم الاقتصاد والهندسة وعمليات البحث. تلعب أساليب البرمجة الرياضية دورًا مهمًّا في **التعلُّم العميق** (Deep Learning)، وتمتلك نماذج التعلُّم العميق عددًا كبيرًا من المُعامِلات التي تحتاج أن تتعلّم من البيانات، حيث تُستخدم خوارزميات التحسين لتعديل مُعامِلات النموذج من أجل تقليل دالة التكلفة التي تقيس الفَرق بين مُخرَجات النموذج المتنبّأ بها والمُخرَجات الصحيحة. تم تطوير العديد من خوارزميات التحسين الخاصة بنماذج التعلُّم العميق مثل: خوارزمية آدم (Adam)، وخوارزمية **الاشتقاق التكيُّفي** (AdaGrad)، وخوارزمية **نشر متوسط الجذر التربيعي** (RMSprop).

**+ الإيجابيات**

تتعامل البرمجة الرياضية مع مجموعة واسعة من مشكلات التحسين وهي غالبًا تضمن الوصول إلى الحلّ الأمثل.

**- السلبيات**

يُعدُّ كلٌّ من التكلفة الحاسوبية للمشكلات الكبيرة وتعقيد إنشاء الصيغة الرياضية المناسبة مرتفعَين بالنسبة لمشكلات العالم الواقعي المعقدة.

## مثال عملي: تحسين مشكلة تشكيل الفريق
## A Working Example: Optimization for the Team-Formation Problem

سيوضِّح هذا الدرس استخدام **خوارزمية القوة المُفرطة** (Brute-Force Algorithm)، **والخوارزمية الاستدلالية الجشعة** (Greedy Heuristic Algorithm) لحلّ مشكلة اتخاذ القرار المُرتكزة على مشكلة تخصيص الموارد القائمة على الفريق والتي تم وصفها سابقًا، بعد ذلك ستتم مقارنة نتائج هاتين الخوارزميتين.

يُمكن استخدام الدالة التالية لإنشاء أمثلة عشوائية لمشكلة تشكيل الفِرق، وتسمح هذه الدالة للمستخدِم أن يُحدِّد أربعة مُعامِلات هي: العدد الإجمالي للمهارات التي يجب أن تؤخذ بعين الاعتبار، والعدد الإجمالي للعمّال المتوفّرين، وعدد المهارات التي يجب أن تتوفّر في أعضاء الفريق بشكل جماعي حتى ينجزوا المُهِمَّة، والعدد الأقصى للمهارات التي يُمكن أن يمتلكها كل عامل.

وبعد ذلك، تقوم الدالة بإنشاء وإظهار مجموعة عمّال لديهم عدة مهارات مُختلفة، وعدة مهارات مطلوبة، وتَستخدم هذه الدالة المكتبة الشهيرة Random التي يُمكن استخدامها في إخراج عيّنة أعداد عشوائية من مجموعة أعداد معيّنة أو عناصر عشوائية من قائمة معيّنة.

**الخوارزمية الاستدلالية الجشعة (Greedy Heuristic Algorithm):**

هي أسلوب استدلالي لحلّ المشكلات، وفيه تقوم الخوارزمية ببناء الحلّ خطوةً خطوةً، وتختار الخيار الأمثل محليًّا في كل مرحلة، حتى تصل في النهاية إلى حلّ شامل ونهائي.

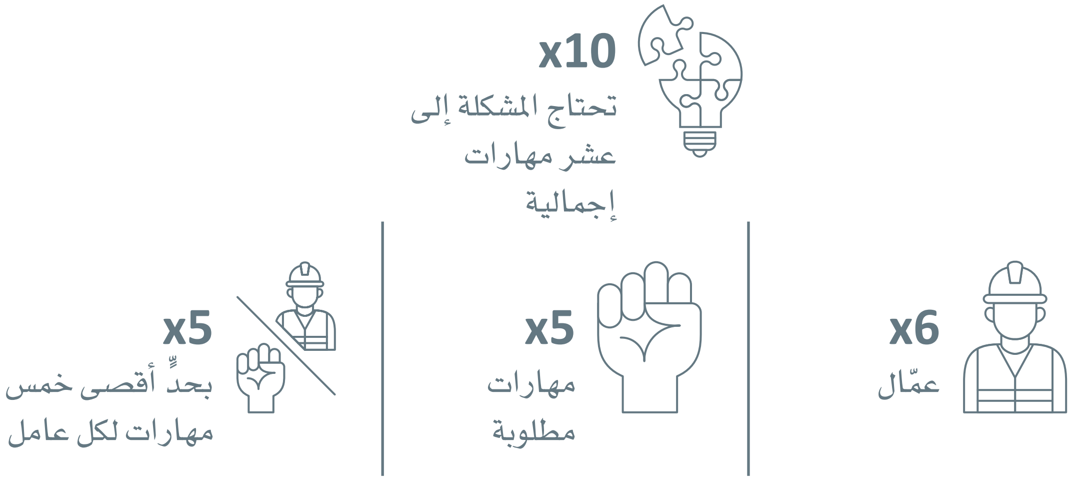
```
import random

def create_problem_instance(skill_number, # total number of skills
                            worker_number, # total number of workers
                            required_skill_number, # number of skills the team has to cover
                            max_skills_per_worker # max number of skills per worker
                             ):
```



وزارة التعليم
Ministry of Education
2025 - 1447

```python
# creates the global list of skills s1, s2, s3, ...
skills = ['s' + str(i) for i in range(1, skill_number+1)]

worker_skills = dict() # dictionary that maps each worker to their set of skills

for i in range(1, worker_number+1): # for each worker

    # makes a worker id (w1, w2, w3, ...)
    worker_id = 'w' + str(i)

    # randomly decides the number of skills that this worker should have (at least 1)
    my_skill_number = random.randint(1, max_skills_per_worker)

    # samples the decided number of skills
    my_skills = set(random.sample(skills, my_skill_number))

    # remembers the skill sampled for this worker
    worker_skills[worker_id] = my_skills

# randomly samples the set of required skills that the team has to cover
required_skills = set(random.sample(skills, required_skill_number))

# returns the worker and required skills
return {'worker_skills':worker_skills, 'required_skills':required_skills}
```

ستقوم الآن باختبار الدالة الواردة سابقًا من خلال إنشاء نسخة من مشكلة معطياتها كالتالي: عشر مهارات إجمالية، وستة عمّال، وتتطلب خمس مهارات كحدٍّ أقصى لكل عامل.




شكل 5.2: رسم توضيحي للمثال الخاص بالمشكلة

**بسبب الطبيعة العشوائية للدالة، ستحصل على نسخة مختلفة من المشكلة في كل مرة تقوم فيها بتشغيل هذا المقطع البرمجي.**

```python
# the following code represents the above test
sample_problem = create_problem_instance(10, 6, 5, 5)

# prints the skills for each worker
for worker_id in sample_problem['worker_skills']:
    print(worker_id, sample_problem['worker_skills'][worker_id])

print()

# prints the required skills that the team has to cover
print('Required Skills:', sample_problem['required_skills'])
```

```
w1 {'s10'}
w2 {'s2', 's8', 's5', 's6'}
w3 {'s7', 's2', 's4', 's5', 's1'}
w4 {'s9', 's4'}
w5 {'s7', 's4'}
w6 {'s7', 's10'}

Required Skills: {'s6', 's8', 's7', 's5', 's9'}
```

تتمثل الخطوة التالية في إنشاء **خوارزمية حلّ (Solver)**، وهي خوارزمية تحسين يُمكنها أن تحدِّد أقل عدد ممكن لفريق العمّال الذي يُمكن اعتماده لإستيفاء كل المهارات المطلوبة.

## اتخاذ القرار بخوارزمية القوة المُفرطة
## Decision Making with a Brute-Force Algorithm

ستُطبِّق أول خوارزمية حلّ أسلوب القوة المُفرطة الذي يعتمد على التعداد الشامل لكل الفِرق المُمكنة وأخذها بعين الاعتبار، وسَتَستخدم هذه الخوارزمية أدوات combinations (توافيق) من وحدة itertools؛ لتوليد كل الفِرق المُمكنة ذات العدد المحدَّد.

سيتم توضيح الأداة بالمثال البسيط أدناه:

```python
# used to generate all possible combinations in a given list of elements
from itertools import combinations

L = ['w1', 'w2', 'w3', 'w4']

print('pairs', list(combinations(L, 2))) # all possible pairs
print('triplets', list(combinations(L, 3))) # all possible triplets
```

```
pairs [('w1', 'w2'), ('w1', 'w3'), ('w1', 'w4'), ('w2', 'w3'), ('w2',
'w4'), ('w3', 'w4')]
triplets [('w1', 'w2', 'w3'), ('w1', 'w2', 'w4'), ('w1', 'w3', 'w4'),
('w2', 'w3', 'w4')]
```


وزارة التعليم
Ministry of Education
2025 - 1447

بعد ذلك، يُمكن إنشاء الدالة التالية لحلّ مشكلة تكوين الفريق بأسلوب القوة المُفرطة، وهذه الخوارزمية تأخذ بعين الاعتبار جميع أحجام الفرق الممكنة، وتنشىء الفِرق بناءً على الأعداد الممكنة، ثم تحصر الفِرق التي تستوفي كل المهارات المطلوبة وتحدِّد الفريق الأقل عددًا:

```
def brute_force_solver(problem):

    worker_skills = problem['worker_skills']
    required_skills = problem['required_skills']

    worker_ids = list(worker_skills.keys()) # gets the ids of all the workers
    worker_num = len(worker_ids) # total number of workers
    all_possible_teams = [] # remembers all possible teams
    best_team = None # remembers the best (smallest) team found so far

    #for each possible team size (singles, pairs, triplets, ...)
    for team_size in range(1, worker_num+1):

        # creates all possible teams of this size
        teams = combinations(worker_ids, team_size)
        for team in teams: # for each team of this size

            skill_union = set() # union of skills covered by all members of this team
            for worker_id in team: # for each team member
                # adds their skills to the union
                skill_union.update(worker_skills[worker_id])

            # if all the required skills are included in the union
            if required_skills.issubset(skill_union):

                # if this is the first team that covers all required skills
                # or this team is smaller than the best one or
                if best_team == None or len(team) < len(best_team):
                    best_team = team # makes this team the best one

    return best_team # returns the best solution
```

من الممكن ألا يكون هناك حلّ لنسخة المشكلة الواردة، فإذا كانت مجموعة المهارات المطلوبة تشمل مهارة لا يمتلكها أي عامل من العمّال المتواجدين، فلن تجد طريقة لإنشاء فريق يغطي كل المهارات، وفي مثل هذه الحالات ستُظهر الخوارزمية المذكورة سابقًا النتيجة بعدم وجود حلّ.

يُمكنك الآن استخدام المقطع البرمجي التالي لاختبار خوارزمية الحلّ بالقوة المُفرطة وفقًا للمثال الذي تم إنشاؤه سابقًا:

```
# uses the brute-force solver to find the best team for the sample problem
best_team = brute_force_solver(sample_problem)
print(best_team)
```

```
('w2', 'w3', 'w4')
```

من المؤكد أن خوارزمية الحلّ بالقوة المُفرطة ستجد أفضل حلّ ممكن، أي: أقلّ الفِرق عددًا طالما أن هناك حلٌّ ممكنٌ، ولكن كما تم مناقشته في بداية هذا الدرس فإن طبيعة الخوارزمية الشمولية تُؤدي إلى زيادة هائلة في التكلفة الحاسوبية كلما زاد حجم المشكلة.

يُمكن توضيح ذلك من خلال إنشاء نُسخ لمشكلات متعددة من حيث تزايد عدد العمّال، ويُمكن استخدام المقطع البرمجي التالي لتوليد نُسخ متنوعة من مشكلة تكوين الفريق، حيث يتنوع عدد العمّال ليكون: 5 و10 و15 و20، ثم يتم توليد 100 نسخة بعدد العمّال، وتشمل كل النُسخ المهارات الإجمالية العشر، والمهارات الثمان المطلوبة، والخمس مهارات كحدٍّ أقصى لكل عامل:

```
problems_with_5_workers = [] # 5 workers
problems_with_10_workers = [] # 10 workers
problems_with_15_workers = [] # 15 workers
problems_with_20_workers = [] # 20 workers

for i in range(100): # repeat 100 times

    problems_with_5_workers.append(create_problem_instance(10, 5, 8, 5))
    problems_with_10_workers.append(create_problem_instance(10, 10, 8, 5))
    problems_with_15_workers.append(create_problem_instance(10, 15, 8, 5))
    problems_with_20_workers.append(create_problem_instance(10, 20, 8, 5))
```

تَقبل الدالة التالية قائمة بنُسخ المشكلة وخوارزمية الحلّ بالقوة المُفرطة، وتُستخدم هذه الخوارزمية لإجراء العمليات الحسابية ثم استخراج الحلّ لجميع النُسخ، كما أنها تُسجل الوقت الإجمالي المطلوب (بالثواني) لحساب الحلول وكذلك العدد الإجمالي للنُسخ التي يُمكن إيجاد حلّ منها:

```
import time

def gets_solutions(problems,solver):

    total_seconds = 0 # total seconds required to solve all problems with this solver
    total_solved = 0 # total number of problems for which the solver found a solution
    solutions = [] # solutions returned by the solver

    for problem in problems:

        start_time = time.time() # starts the timer
        best_team = solver(problem) # computes the solution
        end_time = time.time() # stops the timer
        solutions.append(best_team) # rememberσ the solution
        total_seconds += end_time-start_time # computes total elapsed time

        if best_team != None: # if the best team is a valid team
            total_solved += 1
    print("Solved {} problems in {} seconds".format(total_solved,
                                                    total_seconds))

    return solutions
```


وزارة التعليم
Ministry of Education
2025 - 1447

يستخدِم المقطع البرمجي التالي هذه الدالة وخوارزمية الحلّ بالقوة المُفرطة لحساب الحلول المُمكنة لمجموعات البيانات التي تم إنشاؤها سابقًا والمُكوَّنة من 5-workers (خمسة_عمّال)، و10-workers (عشرة_عمّال)، و15-workers (خمسة عشر_عاملًا)، و20-workers (عشرين_عاملًا):

```
brute_solutions_5 = gets_solutions(problems_with_5_workers,
          solver = brute_force_solver)

brute_solutions_10 = gets_solutions(problems_with_10_workers,
          solver = brute_force_solver)

brute_solutions_15 = gets_solutions(problems_with_15_workers,
          solver = brute_force_solver)

brute_solutions_20 = gets_solutions(problems_with_20_workers,
          solver = brute_force_solver)
```

```
Solved 23 problems in 0.0019948482513427734 seconds
Solved 80 problems in 0.06984829902648926 seconds
Solved 94 problems in 2.754629373550415 seconds
Solved 99 problems in 109.11902689933777 seconds
```

على الرغم من أن الأعداد المطلوبة سُجلت بواسطة الدالة ( )gets_solutions إلا أنها ستكون متفاوتة نظرًا للطبيعة العشوائية لمجموعات البيانات، وسيكون هناك نمطان ثابتان على الدوام هما:

- زيادة عدد العمّال تُؤدي إلى عدد أكبر من نُسخ المشكلات التي من الممكن إيجاد حلّ لها، وهذا النمط من الحلول معقول ومتوقَّع؛ لأن وجود عدد كبير من العمّال يزيد من احتمال وجود عاملٍ واحدٍ على الأقل يمتلك مهارة واحدة مطلوبة ضمن مجموعة العمّال المتاحة.
- زيادة عدد العمّال يؤدي إلى زيادة كبيرة (أُسِّيَّة) في الزمن الحاسوبي، وهذا متوقع حسب التحليل الذي تم إجراؤه في بداية هذا الدرس، وبالنسبة لمجموع العمّال ممن هم بعدد: خمسة، وعشرة، وخمسة عشر، وعشرون عاملًا، فإن عدد الفِرق المُمكنة يساوي: 31، 1023، 32767، و1048575 على الترتيب.

بصفة عامة، وبالنظر إلى عدد العمّال المُعطى N، فإن عدد الفِرق المُمكنة يساوي $2^N-1$، وهذا العدد سيصبح كبيرًا لتقييمه حتى بالنسبة للقيم الصغيرة لـ N. كذلك بالنسبة لأي مشكلة بسيطة بها قيد واحد (يغطي جميع المهارات المطلوبة) وهدف واحد (تقليل حجم الفريق)، فإن القوة المُفرطة قابلة للتطبيق فقط على مجموعات البيانات الصغيرة جدًا، وذلك بالتأكيد ليس حلًا عمليًّا لأي من مشكلات التحسين المعقدة التي نواجها في الواقع والتي أشرنا إليها في بداية هذا الدرس.

## اتخاذ القرار باستخدام خوارزمية استدلالية جشعة
## Decision Making with a Greedy Heuristic Algorithm

تتعامل الدالة التالية مع هذا القيد بواسطة تنفيذ خوارزمية تحسين تعتمد على الأسلوب الاستدلالي الجشع، حيث تقوم الخوارزمية تدريجيًا بتكوين الفريق عن طريق إضافة عضو واحد في كل مرة، فالعضو الذي أضيف مؤخرًا يكون دائمًا هو العضو الذي يمتلك معظم المهارات التي لم توجد في سابقه، وتستمر العملية حتى تستوفي جميع المهارات المطلوبة.

**الدالة الاستدلالية الجشعة (Greedy Heuristic) المُستخدَمة في هذا المثال هي معيار لاختيار عامل يتوفر فيه أكبر عدد من المهارات التي تُستوفى في الفريق إلى الآن، ويمكن استخدام دالة استدلالية أخرى، مبنية على إضافة العامل الذي يتوفر فيه العدد الأكبر من المهارات أولًا.**

```python
def greedy_solver(problem):

    worker_skills = problem['worker_skills']
    required_skills = problem['required_skills']

    # skills that still have not been covered
    uncovered_required_skills = required_skills.copy()
    best_team = []
    # remembers only the skills of each worker that are required but haven't been covered yet
    uncovered_worker_skills = {}

    for worker_id in worker_skills:

        # remembers only the required uncovered skills that this worker has
        uncovered_worker_skills[worker_id] = worker_skills[worker_id].
intersection(uncovered_required_skills)

    # while there are still required skills to cover
    while len(uncovered_required_skills) > 0:

        best_worker_id = None  # the best worker to add next
        # number of uncovered skills required for the best worker to cover
        best_new_coverage = 0

        for worker_id in uncovered_worker_skills:

            # uncovered required skills that this worker can cover
            my_uncovered_skills = uncovered_worker_skills[worker_id]

            # if this worker can cover more uncovered required skills than the best worker so far
            if  len(my_uncovered_skills) > best_new_coverage:
                best_worker_id=worker_id # makes this worker the best worker
                best_new_coverage=len(my_uncovered_skills)

        if best_worker_id != None: # if a best worker was found
            best_team.append(best_worker_id) # adds the worker to the solution

            #removes the best worker's skills from the skills to be covered
            uncovered_required_skills = uncovered_required_skills -
                                        uncovered_worker_skills[best_worker_id]

            for worker_id in uncovered_worker_skills:

                # remembers only the required uncovered skills that this worker has
                uncovered_worker_skills[worker_id] =
uncovered_worker_skills[worker_id].intersection(uncovered_required_skills)

        else: # no best worker has been found and some required skills are still uncovered
            return None  # no solution could be found

    return best_team
```

تُظهر الدالة ()intersections مجموعة جديدة تحتوي فقط على المهارات المشتركة من جميع مهارات العمّال الموجودة في worker_skills، والمهارات المطلوبة التي لم تُستوفَ في .uncovered_worker_skills


وزارة التعليم
Ministry of Education
2025 - 1447

لا تأخذ خوارزمية الحلّ الجشعة كل الفِرق المُمكنة بعين الاعتبار ولا تضمن إيجاد الحلّ الأمثل، ولكنها كما هو موضَّح أدناه أسرع بكثير من خوارزمية الحلّ التي تعتمد على القوة المُفرطة، ومع ذلك يُمكنها أن تُنتج حلولًا جيدة، هي في الغالب حلولٌ مثلى، ومن المؤكد أن تجد هذه الطريقة حلًّا إذا كان موجودًا.

يستخدِم المقطع البرمجي التالي خوارزمية الحلّ الجشعة لحساب حلول مجموعات البيانات: 5-workers (خمسة_عمّال)، و10-workers (عشرة_عمّال)، و15-workers (خمسة عشر_عاملًا)، و20-workers (عشرين_عاملًا) التي تم استخدامها سابقًا لتقييم خوارزمية الحلّ بالقوة المُفرطة:

```
greedy_solutions_5 = gets_solutions(problems_with_5_workers,
           solver = greedy_solver)

greedy_solutions_10 = gets_solutions(problems_with_10_workers,
           solver = greedy_solver)

greedy_solutions_15 = gets_solutions(problems_with_15_workers,
           solver = greedy_solver)

greedy_solutions_20 = gets_solutions(problems_with_20_workers,
           solver = greedy_solver)
```

```
Solved 23 problems in 0.0009970664978027344 seconds
Solved 80 problems in 0.000997304916381836 seconds
Solved 94 problems in 0.001995086669921875 seconds
Solved 99 problems in 0.0019943714141845703 seconds
```

والآن يتضح الفَرق في السرعة بين الخوارزميتين؛ حيث يُمكن تطبيق خوارزمية الحلّ الجشعة على النُسخ المتعلقة بالمشكلات الكبيرة جدًا، كما في المثال التالي:

```
# creates 100 problem instances of a team formation problem with 1000 workers
problems_with_1000_workers = []

for i in range(100): # repeats 100 times
    problems_with_1000_workers.append(create_problem_instance(10, 1000, 8, 5))

# solves the 100-worker problems using the greedy solver
greedy_solutions_1000 = gets_solutions(problems_with_1000_workers,
           solver = greedy_solver)
```

```
Solved 100 problems in 0.09574556350708008 seconds
```

## مقارنة الخوارزميات Comparing the Algorithms

بعد أن تم توضيح ميزة السرعة لخوارزمية الحلّ الاستدلالية الجشعة، تتمثّل الخطوة التالية في التحقق من جودة الحلول التي تُنتجها، حيث تَقبل الدالة التالية الحلول التي أنتجتها الخوارزمية الجشعة وخوارزمية القوة المُفرطة على نفس مجموعة نُسخ المشكلات، ثم تبيّن النِّسب المئوية للنُسخ التي تقوم كلتا الخوارزميتين بذكر الحلّ الأمثل لها (الفريق الأقل عددًا):

```
def compare(brute_solutions,greedy_solutions):
    total_solved = 0
    same_size = 0

    for i in range(len(brute_solutions)):

        if brute_solutions[i] != None: # if a solution was found
            total_solved += 1

            # if the solvers reported a solution of the same size
            if len(brute_solutions[i]) == len(greedy_solutions[i]):
                same_size += 1

    return round(same_size / total_solved, 2)
```

يُمكن الآن استخدام الدالة ()compare لمقارنة فاعلية الخوارزميتين المطبقتين على: الخمسة عمّال، والعشرة عمّال، والخمسة عشر عاملًا، والعشرين عاملًا.

```
print(compare(brute_solutions_5,greedy_solutions_5))
print(compare(brute_solutions_10,greedy_solutions_10))
print(compare(brute_solutions_15,greedy_solutions_15))
print(compare(brute_solutions_20,greedy_solutions_20))
```

```
1.0
0.82
0.88
0.85
```

توضِّح النتائج أن الخوارزمية الاستدلالية الجشعة يُمكنها أن تجد باستمرار الحلّ الأمثل لحوالي 80 % أو أكثر من كل نُسخ المشكلات القابلة للحلّ. وفي الواقع، يُمكن التحقق بسهولة من أن حجم الفريق الذي تُنتجه الخوارزمية الاستدلالية الجشعة حتى في النُسخ التي تفشل في إيجاد الحلول المُثلى لها يكون قريبًا جدًا من حجم أفضل فريق ممكن.

إذا تمت إضافة ذلك إلى ميزة السرعة الهائلة، تجد أن الخوارزمية الاستدلالية خيار عمليّ أكثر للتطبيقات الواقعية، وستكتشف في الدرس التالي تقنيات تحسين أكثر ذكاءً، وستتعرّف على كيفية تطبيقها على مشكلات مُختلفة.


وزارة التعليم
Ministry of Education
2025 - 1447

# تمرينات

1 ما مزايا وعيوب استخدام كلٍّ من: خوارزمية القوة المفرطة والخوارزمية الاستدلالية الجشعة في حلّ مشكلات التحسين؟

2 حلِّلْ طريقة استخدام الخوارزميات الاستدلالية الجشعة لإيجاد الحلول المُثلى في مشكلات التحسين.

وزارة التعليم
Ministry of Education
2025 - 1447

**3** أنشئ خوارزمية حلّ جشعة لتحسين مشكلة تكوين أعضاء فريق، من خلال إكمال المقطع البرمجي التالي بحيث تستخدِم خوارزمية الحلّ الاستدلالية الجشعة لتكليف أعضاء الفريق بالمُهِمَّة:

```
def greedy_solver(problem):
    worker_skills=problem['worker_skills'] # worker skills for this problem
    required_skills=problem['required_skills'] # required skills for this problem

    uncovered_required_skills = required_skills.____________() # skills not covered
    best_team=[] # best solution
    uncovered_worker_skills={}
    for worker_id in worker_skills:
        uncovered_worker_skills[worker_id]=worker_skills[worker_id]._____
(uncovered_required_skills)
    while len(uncovered_required_skills) > 0:
        best_worker_id=__________ # the best worker to add next
        best_new_coverage=0 # number of uncovered required skills covered by the best worker
        for worker_id in uncovered_worker_skills: # for each worker
            my_uncovered_skills=uncovered_worker_skills[worker_id]
            # if this worker can cover more uncovered required skills than the best worker so far
            if  len(my_uncovered_skills)>best_new_coverage:
                best_worker_id=worker_id # makes this worker the best worker

                best_new_coverage=____________(my_uncovered_skills)

        if best_worker_id!=_____________: # if a best worker was found

            best_team.___________(best_worker_id) # adds the worker to the solution
            #removes the best worker's skills from the skills to be covered
            uncovered_required_skills=uncovered_required_skills - uncovered_
worker_skills[best_worker_id]
            # for each worker
            for worker_id in uncovered_worker_skills:

                # remembers only the required uncovered skills that this worker has

                uncovered_worker_skills[worker_id]=uncovered_worker_
skills[worker_id].________(uncovered_required_skills)
        else: # no best worker has been found and some required skills are still uncovered

            return _______ # no solution could be found
    return best_team
```

**4** اذكر ثلاث مشكلات تحسين مُختلفة من العالم الواقعي، وفي كل مشكلة:

- اضرب مثالًا على دالة موضوعية.
- اضرب مثالين على القيود إن وُجِدَتْ.

**5** إذا قمتَ بزيادة عدد العمّال في خوارزمية القوة المُفرطة، كيف يؤثر ذلك على المشكلة من حيث عدد الحلول والزمن الحسابي؟

وزارة التعليم
Ministry of Education
2025 - 1447

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الثاني

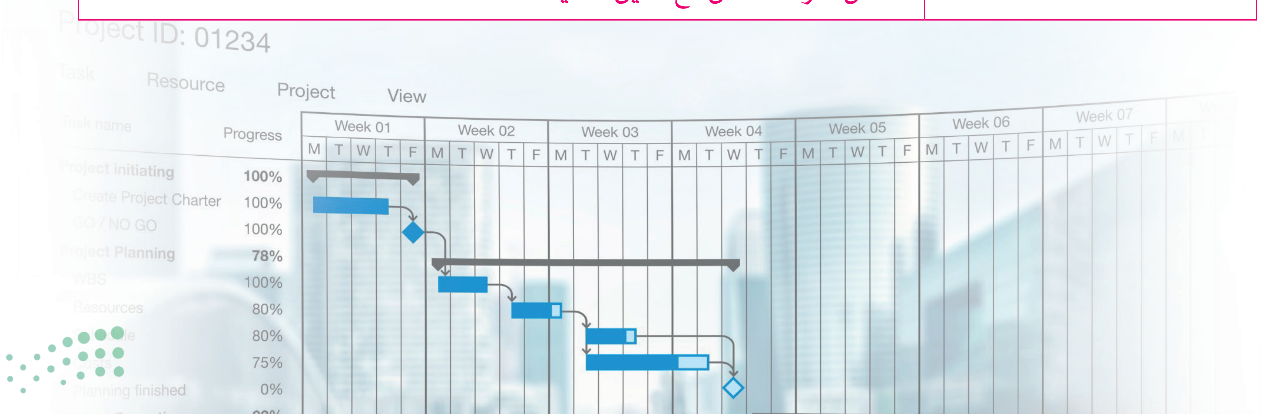
# مشكلة جدولة الموارد

## مشكلات الجدولة Scheduling Problems

مشكلات الجدولة شائعة في مجال التحسين؛ لأنها تتطلب تخصيص موارد محدودة لمهام متعددة بطريقة تُحسِّن بعض الدوال الموضوعية، وعادة ما تكون لمشكلات الجدولة قيود إضافية مثل: الحاجة إلى تنفيذ المهام بترتيب معيّن أو إنجازها في الموعد النهائي المحدَّد، وهذه المشكلات جوهرية في العديد من المجالات المختلفة بما فيها التصنيع والنقل والرعاية الصحية وإدارة المشاريع. ستتعمق في هذا الدرس في خوارزميات التحسين عن طريق إدخال تقنيات إضافية لحلّ جدولة المشكلات.

**جدول 5.1: تطبيقات من مجالات مختلفة بحاجة إلى حلول الجدولة**

| | |
|---|---|
| جدولة المشاريع | تخصيص الموارد والمهام لأنشطة المشروع؛ لتقليل مدة المشروع وتكاليفه. |
| تخطيط الإنتاج | تحديد خطة الإنتاج المُثلى؛ لتلبية الطلب مع تقليل المخزون والتكاليف. |
| جدولة خطوط الطيران | جدولة إقلاع الطائرات وفترات عمل الطاقم؛ لتحسين جداول الرحلات مع تقليل التأخير والتكاليف. |
| جدولة مركز الاتصالات | تخصيص فترات عمل للموظفين؛ لضمان التغطية المناسبة لفترات العمل مع تقليل التكاليف والالتزام باتفاقيات مستوى الخدمة. |
| جدولة الإنتاج حسب الطلب | تخصيص الموارد في التصنيع؛ لتقليل زمن الإنتاج والتكاليف. |
| جدولة وسائل الإعلام | جدولة توقيت الإعلانات على التلفاز أو الإذاعة؛ لزيادة الوصول إلى الجمهور والإيرادات مع الالتزام بقيود الميزانية. |
| جدولة الممرضات | تخصيص فترات عمل للممرضات في المستشفيات؛ لضمان التغطية الكافية خلال فترات العمل مع تقليل تكاليف العمالة. |




شكل 5.3: مُخطّط قانت يبيّن جدول مشروع


وزارة التعليم
Ministry of Education
2025 - 1447

في هذا الدرس ستُستخدم مشكلة **التباطُؤ الموزون للآلة الواحدة** (Single-Machine Weighted Tardiness - SMWT) كمثال عملي لتوضيح كيف يُمكن لخوارزميات التحسين أن تحلّ مشكلات الجدولة.

## مشكلة التباطُؤ الموزون للآلة الواحدة
## Single-Machine Weighted Tardiness (SMWT) Problem

لتوضيح هذه المشكلة، سنفترض أن مَصنعًا يرغبُ في جدولة مهام إنتاج عدة سلع على آلة واحدة، على النحو التالي:

- كل مُهِمَّة لها وقت معالجة محدَّد، وموعد محدَّد لابد أن تكتمل فيه.
- كل مُهِمَّة مرتبطة بوزن يمثل أهميتها.

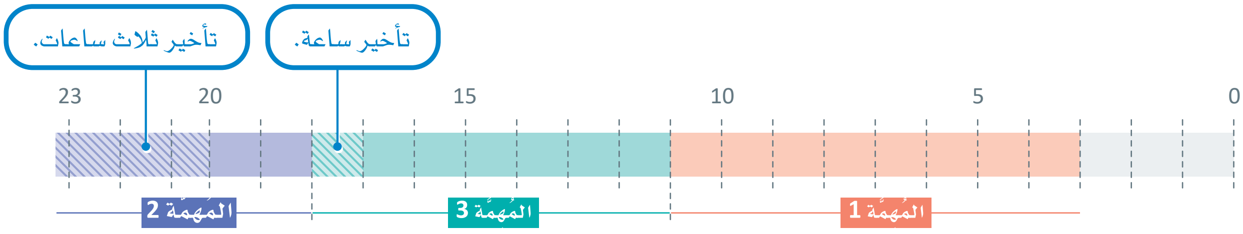
إذا كان من المستحيل إنجاز كل المهام في الموعد النهائي، فسيكون عدم الالتزام بإنجاز المهام ذات الوزن الصغير في الموعد النهائي أقل تكلفة من عدم الالتزام بإنجاز المهام ذات الوزن الكبير في الموعد النهائي.

### الهدف

**الهدف** (Goal) من جدولة المهام بطريقة محدَّدة هو تقليل المجموع الموزون للتأخير (التباطُؤ) لكل مُهِمَّة، وهكذا فإن مجموع التباطُؤ الموزون يكون بمثابة الدالة الموضوعية لخوارزميات التحسين المصمَّمة لحلّ هذه المشكلة.

### حساب التأخير

يُحسبُ **التأخير** (Lateness) في أداء المُهِمَّة على أساس الفَرق بين زمن إنجازها والموعد المحدَّد لتسليمها، ثم تُستخدم أوزان المهام **كعوامل ضرب** (Multipliers) لإكمال المجموع الموزون النهائي. على سبيل المثال: افترض أن هناك جدولًا به ثلاث مهام هي: م1 وم2 وم3، وأوزان هذه المهام هي: 2 و1 و2 على الترتيب. وفقًا لهذا الجدول، ستُنجز المُهِمَّة رقم 1 في الموعد المحدَّد، وسيتأخر إنجاز المُهِمَّة رقم 2 ثلاث ساعات عن موعد تسليمها، أما المُهِمَّة رقم 3 فسيتأخر إنجازها ساعة واحدة عن موعد تسليمها، ويعني ذلك أن مجموع التباطُؤ الموزون يساوي 5 = 2×1+1×3.




شكل 5.4: رسم توضيحي لتسلسل المهام

| المُهِمَّة | الموعد المحدَّد لإنجازها | موعد تسليمها | التأخير | التباطُؤ الموزون |
|---|---|---|---|---|
| م1 | 14 | 11 | 0 | 0 |
| م2 | 20 | 23 | 3 | 3 |
| م3 | 17 | 18 | 1 | 2 |

شكل 5.5: حساب التباطُؤ الموزون

توجد صعوبة في حلّ مشكلة التباطُؤ الموزون للآلة الواحدة؛ لأن تعقُّدَها يتزايد تزايُدًا أُسِّيًّا مع عدد المهام، مما يجعل إيجاد أفضل حلّ ممكن لأحجام المُدخَلات الكبيرة مكلفًا للغاية وعادة ما يكون مستحيلًا.

**تُستخدم خوارزميات التحسين للحصول على حلول شبه مثالية لمشكلة محدَّدة في مدة زمنية معقولة.**

وزارة التعليم
Ministry of Education
2025 - 1447

## مشكلة جدولة الإنتاج حسب الطلب Job Shop Scheduling (JSS) Problem

مشكلة **جدولة الإنتاج حسب الطلب** (JSS) هي مشكلة اعتيادية أخرى في الجدولة حَظِيت بدراسات مُوسَّعة في مجال التحسين، وتتضمن جدولة مجموعة من المهام على عدة آلات، حيث يجب معالجة كل مُهِمَّة بترتيب ووقت معيّنان لكل آلة بالنسبة للمهام الأخرى.

**الهدف**

تقليل زمن الإنجاز الكليّ (فترة التصنيع) لجميع المهام.

**متغيِّرات المشكلة**

المتغيِّرات الأخرى من هذه المشكلة تفرض عدة قيود إضافية مثل:

- وجوب الالتزام بتاريخ إصدار كل مُهِمَّة؛ حيث إن لكل مُهِمَّة تاريخها الخاص ولا يمكن البدء بها قبل ذلك التاريخ، بالإضافة إلى مراعاة الموعد النهائي.
- وجوب جدولة بعض المهام قبل المهام الأخرى؛ بسبب ضوابط الأسبقية بينها.
- وجوب إخضاع كل آلة للصيانة الدورية وفقًا لضوابط جدول الصيانة، حيث لا يمكن للآلات تأدية المهام أثناء الصيانة، كما لا يمكن أن تتوقف المُهِمَّة بمجرد بدئها.

لا بد أن تمر كل آلة بفترة توقُّف عن الإنتاج بعد إكمال المُهِمَّة، وقد يكون طول هذه الفترة ثابتًا، وقد يتفاوت من آلة إلى أخرى، ومن الممكن أن يعتمد على الوقت الذي استغرقته الآلة في إكمال المُهِمَّة السابقة.

ما ورد أعلاه ليس سوى مجموعة فرعية من القيود المعقدة والمتعددة، ومن متغيِّرات المشكلة الموجودة في مشكلات الجدولة التي نواجها في واقع الحياة، حيث أن لكل متغيِّر خصائصه وتطبيقاته العملية الفريدة، وقد تكون خوارزميات التحسين المُختلفة أكثر ملاءمة لحلّ كل متغيِّر من متغيِّرات المشكلة.

## استخدام البايثون والتحسين لحلّ مشكلة التباطُؤ الموزون للآلة الواحدة
## Using Python and Optimization to Solve the SMWT Problem

يُمكن استخدام المقطع البرمجي التالي لإنشاء نُسَخ عشوائية لمشكلة **التباطُؤ الموزون للآلة الواحدة** (SMWT):

```
import random

# creates an instance of the Single-Machine Weighted Tardiness problem.

def create_problem_instance(job_num, # number of jobs to create
                            duration_range, # job duration range
                            deadline_range, # deadline range
                            weight_range):# importance weight range

    # generates a random duration, deadline, and weight for each job
    durations = [random.randint(*duration_range) for i in range(job_num)]
    deadlines = [random.randint(*deadline_range) for i in range(job_num)]
    weights = [random.randint(*weight_range) for i in range(job_num)]

    # returns the problem instance as a dictionary
    return {'durations':durations,
            'deadlines':deadlines,
            'weights':weights}
```


وزارة التعليم
Ministry of Education
2025 - 1447

تُستخدم الدالة random.randint(x,y) لتوليد عدد صحيح عشوائي بين x وy، وهناك طريقة مُختلفة لاستخدام هذه الدالة تتمثّل في توفير قائمة [x,y] أو مجموعة (x,y)، وفي هذه الحالة لا بد من كتابة الرمز * قبل القائمة، كما هو موضَّح في الدالة السابقة، على سبيل المثال:

```
for i in range(5):# prints 5 random integers between 1 and 10
    print(random.randint(*[1, 10]))
```

```
6
5
5
10
1
```

يَستخدم المقطع البرمجي التالي دالة create_problem_instance( ) لتوليد نسخة لمشكلة يتوفّر فيها ما يلي:

- تشتمل كلّ نسخة على عشرة مهام.
- يُمكن لكل مُهِمَّة أن تستمر ما بين 5 وحدات زمنية و20 وحدة زمنية، وسيتم افتراض أن الساعة هي الوحدة الزمنية المستخدَمة فيما تبقى من هذا الدرس.
- كل مُهِمَّة لها موعد نهائي يتراوح ما بين 5 ساعات و50 ساعة، وتبدأ ساعة الموعد النهائي من لحظة بدء المُهِمَّة الأولى في استخدام الآلة، على سبيل المثال: إذا كان الموعد النهائي لمُهِمَّة ما يساوي عشر ساعات، فهذا يعني أنه لا بد من إكمال المُهِمَّة في غضون عشر ساعات من بداية المُهِمَّة الأولى في الجدول.
- وزن كل مُهِمَّة هو عدد صحيح يتراوح بين 1 و3.

عدد المهام المراد إنشاؤها.

نطاق مدة المُهِمَّة.

```
create_problem_instance(10, [5, 20], [5, 50], [1, 3])
```

نطاق الموعد النهائي.

مدى أهمية الوزن.

```
{'durations': [18, 17, 17, 6, 9, 6, 20, 12, 9, 19],
 'deadlines': [39, 31, 6, 42, 48, 10, 39, 16, 34, 35],
 'weights': [2, 2, 3, 2, 1, 3, 2, 1, 3, 1]}
```

يُمكن استخدام الدالة التالية لتقييم جودة أي جدول أنتجته إحدى الخوارزميات لنسخة مشكلة محدَّدة، حيث تَقبل الدالة نسخة المشكلة وجدولًا لمهامها، ثم تمر على كل المهام بترتيب جدولتها نفسه حتى تَحسب أزمنة إنجازها ومجموع التباطُؤ الموزون لكامل الجدول، ويُحسب هذا التباطُؤ بحساب تباطؤ كل مُهِمَّة (مع مراعاة الموعد النهائي لها) وضربه في وزن المُهِمَّة وإضافة الناتج إلى المجموع:

```
# computes the total weighted tardiness of a given schedule for a given problem instance

def compute_schedule_tardiness(problem, schedule):

    # gets the information for this problem
    durations, weights, deadlines=problem['durations'], problem['weights'],
problem['deadlines']

    job_num = len(schedule) # gets the number of jobs
    finish_times = [0] * job_num # stores the finish time for each job
    schedule_tardiness = 0 # initializes the weighted tardiness of the overall schedule to 0
    for pos in range(job_num): # goes over the jobs in scheduled order
```

```
        job_id=schedule[pos]  # schedule[pos] is the id in the 'pos' position of the schedule

        if pos == 0:  # if this is the job that was scheduled first (position 0)

            # the finish time of the job that starts first is equal to its run time
            finish_times[pos] = durations[job_id]

        else:  # for all jobs except the one that was scheduled first

            # the finish time is equal to the finish time of the previous time plus the job's run time
            finish_times[pos] = finish_times[pos-1] + durations[job_id]

        # computes the weighted tardiness of this job and adds it to the schedule's overall tardiness
        schedule_tardiness += weights[job_id] * max(finish_times[pos] -
deadlines[job_id], 0)

    return schedule_tardiness,finish_times
```

ستُستخدم الدالة ( )compute_schedule_tardiness لتقييم الجداول، وستكون هذه الدالة بمثابة أداة مفيدة لكل الخوارزميات التي سيتم تقديمها في هذا الدرس لحلّ مشكلة **التباطؤ الموزون للآلة الواحدة** (SMWT).

## *fx* دالة التباديل Itertools.Permutations( ) Function

تستخدِم خوارزمية حلّ القوة المُفرطة الدالة ( )itertools.permutations لإنشاء كل الجداول المُمكنة (تجميعات المهام)، ثم تَحسب تباطؤ كل جدول ممكن وتستخرج أفضل جدول (الجدول ذو التباطؤ الكُليّ الأدنى).

تَقبل الدالة ( )itertools.permutations عنصرًا واحدًا متكررًا (مثل: قائمة) وتُنشئ كل تبديل ممكن لقيم المُدخلات، ويوضِّح المثال البسيط التالي استخدام دالة ( )permutations ويُظهِر التبديلات لكل عناوين المهام المُعطاة:

تُستخدم خوارزميات حلّ القوة المُفرطة بشكل أفضل لحلّ المشكلات الصغيرة، فالنسخة الخاصة بمشكلة التباطؤ الموزون للآلة الواحدة ذات عدد N من المهام، لديها عدد N! من الجداول المكنة، فعندما يكون N = 5، سيكون الناتج 120 = 5! جدولًا، ولكن هذا العدد يتزايد بشكل كبير عندما يكون N = 10 إلى 3,628,800 = 10! ، وعندما يكون N = 11 إلى 39,916,800 = 11!.

```
job_ids = [0,1,2] # the ids of 3 jobs
for schedule in itertools.permutations(job_ids):
    print(schedule)
```

```
(0, 1, 2)
(0, 2, 1)
(1, 0, 2)
(1, 2, 0)
(2, 0, 1)
(2, 1, 0)
```

## خوارزمية حلّ القوة المُفرطة Brute-Force Solver

لقد تعلّمت في الدرس السابق طريقة استخدام خوارزمية حلّ القوة المُفرطة في مشكلة تكوين فريق، وعلى الرغم من أن خوارزمية الحلّ هذه أظهرت بطئًا شديدًا في المشكلات الأكبر حجمًا، إلا أن قدرتها على إيجاد الحلّ الأمثل (أفضل حلّ ممكن) لنُسخ المشكلة ذات الحجم الصغير كانت مفيدة في تقييم جودة الحلول المُنتَجة بواسطة خوارزميات التحسين الأسرع التي لا تضمن إيجاد الحلّ الأمثل. وبالمثل: يُمكن استخدام خوارزمية حلّ القوة المُفرطة التالية لحلّ مشكلة **التباطؤ الموزون للآلة الواحدة** (SMWT).

```python
import itertools

def brute_force_solver(problem):

    # gets the information for this problem
    durations, weights, deadlines=problem['durations'], problem['weights'],
problem['deadlines']

    job_num = len(durations) # number of jobs

    # Generates all possible schedules
    all_schedules = itertools.permutations(range(job_num))

    # Initializes the best solution and its total weighted tardiness
    best_schedule = None # initialized to None

    # 'inf' stands for 'infnity'. Python will evaluate all numbers as smaller than this value.
    best_tardiness = float('inf')

    # stores the finish time of each job in the best schedule
    best_finish_times = None # initalized to None

    for schedule in all_schedules: # for every possible schedule

        #evalutes the schedule
        tardiness,finish_times=compute_schedule_tardiness(problem, schedule)

        if tardiness < best_tardiness: # this schedule is better than the best so far
            best_tardiness = tardiness
            best_schedule = schedule
            best_finish_times = finish_times

    # returns the results as a dictionary
    return  {'schedule':best_schedule,
             'tardiness':best_tardiness,
             'finish_times':best_finish_times}
```

خوارزمية الحلّ تعطي الجدول الأفضل، وزمن التباطؤ، وزمن إنجاز كل مُهِمَّة مُعطاة في هذا الجدول. على سبيل المثال، إذا كان الجدول يحوي ثلاث مهام، وكانت أوقات إنجاز جميع المهام تساوي [20 ،14 ،10]، فذلك يعني أن المُهِمَّة التي بدأت أولًا انتهت بعد 10 ساعات، والمُهِمَّة الثانية انتهت بعد ذلك بأربع ساعات، والمُهِمَّة الأخيرة انتهت بعد ست ساعات من اكتمال المُهِمَّة الثانية.

عدد المهام المراد إنشاؤها.

نطاق الموعد النهائي.

```python
sample_problem = create_problem_instance(5, [5, 20], [5, 30], [1, 3])
brute_force_solver(sample_problem)
```

نطاق مدة المُهِمَّة.

مدى أهمية الوزن.

```
{'schedule': (0, 2, 1, 3, 4),
 'tardiness': 164,
 'finish_times': [5, 11, 21, 36, 51]}
```

## خوارزمية الحلّ الاستدلالية الجشعة Greedy Heuristic Solver

تستخدِم خوارزمية الحلّ الجشعة أسلوبًا استدلاليًا بسيطًا لفرز المهام واتخاذ قرار الترتيب الذي يجب جدولتها وفقًا له، ثم تُرتب المهام لحساب زمن إكمال كل مُهِمَّة ومجموع التباطُؤ الموزون لكامل الجدول، وفي هذا المثال الخاص تُظهر خوارزمية الحلّ الجشعة نوع المُخرَجات نفسه الذي أظهرته خوارزمية حلّ القوة المُفرطة.

تَقبل خوارزمية الحلّ الجشعة مُعامِلان هما: نسخة المشكلة المراد حلّها، ودالة الاستدلال التي ستُستخدم (معيار فرز المهام)، مما يسمح للمستخدِم بأن يُطبِّق أي دالة استدلال يختارها كدالة البايثون، ثم يمرِّره إلى خوارزمية الحلّ الجشعة باعتباره مُعامِلًا.

تُطبِّق الدالة التالية خوارزمية تحسين تستخدِم دالةً استدلاليةً جشعةً لحلّ المشكلة:

```
def greedy_solver(problem, heuristic):

    # gets the information for this problem
    durations, weights, deadlines = problem['durations'], problem['weights'],
problem['deadlines']

    job_num = len(durations)# gets the number of jobs

    # Creates a list of job indices sorted by their deadline in non-decreasing order
    schedule = sorted(range(job_num), key = lambda j: heuristic(j, problem))


    # evaluates the schedule
    tardiness, finish_times = compute_schedule_tardiness(problem, schedule)

    # returns the results as a dictionary
    return  {'schedule':schedule,
             'tardiness':tardiness,
             'finish_times':finish_times}
```

يُستخدم بناء الجملة lambda مع دالة البايثون ( )sorted عندما يتمثّل الهدف في فرز قائمة عناصر بناءً على قيمة يتم حسابها بطريقة منفصلة لكل عنصر.

يُستخدم في هذا المثال دالة استدلالية جشعة لتحديد المُهِمَّة التالية التي تحتاج إلى جدولة وهي المُهِمَّة التي لها أقرب موعد نهائي.

تُظهر الدالة التالية الموعد النهائي لمُهِمَّة محدَّدة في نسخة مشكلة مُعطاة:

```
# returns the deadline of a given job
def deadline_heuristic(job,problem):

    # accesses the deadlines for this problem and returns the deadline for the job
    return problem['deadlines'][job]
```

تمرير دالة deadline_heuristic كمُعَامِل إلى **خوارزمية الحلّ الجشعة** (greedy_solver) يعني أن الخوارزمية ستُجدوِل (تفرز) المهام وفق ترتيب تصاعدي حسب الموعد النهائي، مما يعني أن المهام التي لها أقرب موعد نهائي ستُجَدوَل أولًا.


وزارة التعليم
Ministry of Education
2025 - 1447

```
greedy_sol = greedy_solver(sample_problem, deadline_heuristic)
greedy_sol
```

```
{'schedule': [3, 1, 4, 0, 2],
 'tardiness': 124,
 'finish_times': [15, 26, 32, 48, 57]}
```

تُطبِّق الدالة التالية استدلالًا بديلًا يأخذ في اعتباره أوزان المهام عند اتخاذ قرار ترتيبها في الجدول:

```
# returns the weighted deadline of a given job
def weighted_deadline_heuristic(job,problem):

    # accesses the deadlines for this problem and returns the deadline for the job
    return problem['deadlines'][job] / problem['weights'][job]
weighted_greedy_sol=greedy_solver(sample_problem, weighted_deadline_heuristic)
weighted_greedy_sol
```

```
{'schedule': [3, 2, 1, 4, 0],
 'tardiness': 89,
 'finish_times': [15, 24, 35, 41, 57]}
```

## البحث المحلّي Local Search

على الرغم من أن خوارزمية الحلّ الجشعة أسرع بكثير من خوارزمية القوة المُفرطة، إلا أنها تميل إلى إنتاج حلول ذات جودة أقل بزمن تباطؤ أعلى، ويُعدُّ البحث المحلّي طريقة لتحسين حلّ تم حسابه بواسطة الخوارزمية الجشعة أو بأي طريقة أخرى.

**البحث المحلّي (Local Search):**
هو طريقة تحسين استدلالية تركِّز على اكتشاف حلول مجاورة لِحلّ معين بهدف تحسينه.

في البحث المحلّي، يُعدَّل الحلّ الذي تم التوصل إليه في البداية بشكلٍ متكرر من خلال فحص الحلول المجاورة التي وُجِدت عن طريق إجراء تعديلات بسيطة على الحلّ الحالي. بالنسبة للعديد من مشكلات التحسين، فهناك طريقة شائعة لتعديل الحلّ تتمثل في تبديل العناصر بشكل متكرر. على سبيل المثال، في مشكلة تكوين الفريق التي تم توضيحها في الدرس السابق، سيحاول أسلوب البحث المحلّي إنشاء فريق أفضل وذلك من خلال تبديل أعضاء الفريق بالعمّال الذين لا يُعدّون حاليًا جزءًا من الفريق.

أنشأت **خوارزمية الحلّ الاستدلالية الجشعة** (Greedy Heuristic Solver) حلًّا للمشكلة خطوة خطوة حتى حصلت في النهاية على حلّ كامل ونهائي، وعلى العكس من ذلك تبدأ طرائق البحث المحلّية بحلّ كامل قد يكون ذا جودة متوسطة أو سيئة، وتعمل بطريقة تكرارية لتحسين جودته. في كل خطوة يكون هناك تغيير بسيط على الحلّ الحالي، وتُقيَّم جودة الحلّ الناتج (يسمّى الحل المُجاور)، وإذا كان يتمتع بجودة أفضل، فإنهُ يستبدل الحلّ الحالي ويستمر في البحث، وإذا لم يكن كذلك، يتم تجاهل الحل المُجاور وتتكرر العملية لتوليد حل مجاور آخر، ثم ينتهي البحث عندما يتعذر العثور على حلّ مُجاور آخر يتمتع بجودة أفضل من الحلّ الحالي، ويتم تحديد أفضل حلّ تم العثور عليه.

## دالة خوارزمية حلّ البحث المحلّي Local_search_solver( ) Function

تطبق الدالة التالية local_search_solver( ) خوارزمية حلّ البحث المحلّي القائم على المُبادلة لِمشكلة **التباطُؤ الموزون للآلة الواحدة** (SMWT)، حيث تَقبل هذه الدالة أربعة مُعامِلات وهي:

- نسخة المشكلة.
- خوارزمية استدلالية جشعة تستخدِمها دالة greedy_solver( ) لحساب حلّ أوّلي.
- دالة swap_selector المستخدَمة لانتقاء مُهمّتين ستتبادلان موقعيهما في الجدول. على سبيل المثال، إذا كان الحلّ الحالي للمشكلة المُكوَّنة من أربع مهام هو [1، 3، 2، 0]، وقرَّرت دالة swap_selector أن يحدث مبادلة بين المُهِمَّة الأولى والمُهمَّة الأخيرة، سيكون الحلّ المرشَّح هو [0، 3، 2، 1].
- max_iterations عدد صحيح يُحدِّد عدد المبادلات التي يجب تجربتها قبل أن تتوصل الخوارزمية للحلّ الأفضل في حينه.

في كل تكرار، تنتقي الخوارزمية مُهمَّتين للتبديل بينهما، ثم تُنشئ جدولًا جديدًا تتم فيه هذه المبادلة، وكل شيء في الجدول الجديد بخلاف ذلك سيكون مُطابقًا للجدول الأصلي. إذا كان للجدول الجديد تباطؤ موزون أقل من الجدول الأفضل الذي تم إيجاده حتى الآن، فإن الجدول الجديد يُصبح هو الأفضل بدلًا منه. خوارزمية الحلّ هذه لها نفس مُخرَجات خوارزمية الحلّ الجشعة وخوارزمية حلّ القوة المُفرطة.

**سلوك خوارزميات التحسين القائمة على البحث المحلي يتأثر بشكل كبير بالاستراتيجية المستخدَمة بطريقة تكرارية لتعديل الحل.**

```
def local_search_solver(problem, greedy_heuristic, swap_selector, max_
iterations):

    # gets the information for this problem
    durations, weights, deadlines=problem['durations'], problem['weights'],
problem['deadlines']

    job_num = len(durations) # gets the number of jobs

    # uses the greedy solver to get a first schedule
    # this schedule will be then iteratively refined through local search
    greedy_sol = greedy_solver(problem, greedy_heuristic) # the best schedule so far

    best_schedule, best_tardiness, best_finish_times = greedy_sol['schedule'],
greedy_sol['tardiness'], greedy_sol['finish_times']

    # local search
    for i in range(max_iterations): # for each of the given iterations

        # chooses which two positions to swap
        pos1, pos2 = swap_selector(best_schedule)

        new_schedule = best_schedule.copy() # create a copy of the schedule

        # swaps jobs at positions pos1 and pos2
        new_schedule[pos1], new_schedule[pos2] = best_schedule[pos2],
                                                 best_schedule[pos1]
```


وزارة التعليم
Ministry of Education
2025 - 1447

```
        # computes the new tardiness after the swap
        new_tardiness, new_finish_times = compute_schedule_tardiness(problem,
new_schedule)

        # if the new schedule is better than the best one so far
        if new_tardiness < best_tardiness:

            # the new_schedule becomes the best one
            best_schedule = new_schedule
            best_tardiness = new_tardiness
            best_finish_times = new_finish_times

    # returns the best solution
    return {'schedule':best_schedule,
            'tardiness':best_tardiness,
            'finish_times':best_finish_times}
```

جيران الحلّ في هذا المثال كلها حلول يتم الحصول عليها عن طريق انتقاء مُهمَّتين داخل الحلّ ومبادلة موقعيهما في الجدول.

تُطبِّق الدالة التالية مبادلة عشوائية بانتقاء مُهمَّتين عشوائيتين في الجدول المُعطى الذي يستوجب تبديل مكانيهما:

```
def random_swap(schedule):

    job_num = len(schedule) # gets the number of scheduled jobs

    pos1 = random.randint(0, job_num - 1) # samples a random position

    pos2 = pos1
    while pos2 == pos1: # keeps sampling until it finds a position other than pos1
        pos2 = random.randint(0, job_num - 1) # samples another random position

    return pos1, pos2 # returns the two positions that should be swapped
```

تستخدِم الدالة التالية استراتيجية مُختلفة وذلك باختيارها الدائم لمُهمَّتين عشوائيتين متجاورتين في الجدول لتبادلهما. على سبيل المثال، إذا كان الجدول الحالي لنسخة مشكلة مُكوَّنة من أربع مهام هو [2 ،1 ،3 ،0]، فإن المبادلات المُرشحة ستكون فقط 0<>3 و3<>1 و1<>2.

```
def adjacent_swap(schedule):

    job_num = len(schedule) # gets the number of scheduled jobs

    pos1 = random.randint(0, job_num - 2) # samples a random position (excluding the last
one)
    pos2 = pos1 + 1 # gets the position after the sampled one

    return pos1,pos2 # returns the two positions that should be swapped
```

يستخدِم المقطع البرمجي التالي استراتيجيتي المبادلة مع خوارزمية حلّ البحث المحلّي لحلّ المشكلة التي تم إنشاؤها في بداية هذا الدرس:

```
print(local_search_solver(sample_problem, weighted_deadline_heuristic, random_
swap, 1000))

print(local_search_solver(sample_problem, weighted_deadline_heuristic,
adjacent_swap, 1000))
```

```
{'schedule': [3, 4, 2, 1, 0], 'tardiness': 83, 'finish_times': [15, 21, 30,
41, 57]}
{'schedule': [3, 4, 2, 1, 0], 'tardiness': 83, 'finish_times': [15, 21, 30,
41, 57]}
```

تُظهر النتائج أفضل جدول [0، 1، 2، 4، 3] لهذا المثال، وإجمالي التباطُؤ 83، وأزمنة إكمال المهام (ستنتهي المُهمَّة 3 في الوحدة 15 من الزمن، وتنتهي المُهمَّة 4 في الوحدة 21 منه، وهكذا).

## مقارنة خوارزميات الحلّ Comparing Solvers

يستخدِم المقطع البرمجي التالي الدالة ()create_problem_instance لتوليد مجموعتي بيانات:

- مجموعة بيانات من 100 نسخة لمشكلة التباطُؤ الموزون للآلة الواحدة، وفي كل منها 7 مهام.
- مجموعة بيانات من 100 نسخة لمشكلة التباطُؤ الموزون للآلة الواحدة، وفي كل منها 30 مُهِمَّة.

سيتم استخدام مجموعة البيانات الأولى لمقارنة أداء جميع خوارزميات الحلّ الموضَّحة في هذا الدرس:

1. خوارزمية حلّ القوة المُفرطة.
2. خوارزمية الحلّ الجشعة المُتضمنة على استدلال خاص بالموعد النهائي.
3. خوارزمية الحلّ الجشعة المُتضمنة على استدلال خاص بالموعد النهائي الموزون.
4. خوارزمية حلّ البحث المحلّي المُتضمنة على مبادلات عشوائية وخوارزمية الحلّ الجشعة ذات استدلال خاص بالموعد النهائي لإيجاد الحلّ الأوّلي.
5. خوارزمية حلّ البحث المحلّي المُتضمنة على مبادلات عشوائية وخوارزمية الحلّ الجشعة ذات استدلال خاص بالموعد النهائي الموزون.
6. خوارزمية حلّ البحث المحلّي المُتضمنة على مبادلات متجاورة وخوارزمية الحلّ الجشعة ذات استدلال خاص بالموعد النهائي.
7. خوارزمية حلّ البحث المحلّي المُتضمنة على مبادلات متجاورة وخوارزمية الحلّ الجشعة ذات استدلال خاص بالموعد النهائي الموزون.

سيتم استخدام مجموعة البيانات الثانية لمقارنة جميع خوارزميات الحلّ باستثناء خوارزمية حلّ القوة المُفرطة البطيئة جدًا بالنسبة للمشكلات المشتملة على 30 مُهِمَّة.

```
#Dataset 1
problems_7 = []
for i in range(100):
    problems_7.append(create_problem_instance(7, [5, 20], [5, 50], [1, 3]))

#Dataset 2
problems_30 = []
for i in range(100):
    problems_30.append(create_problem_instance(30, [5,20], [5, 50], [1, 3]))
```


وزارة التعليم
Ministry of Education
2025 - 1447

## دالة المقارنة Compare( ) Function

تستخدِم الدالـة التاليـة ( )Compare كل خوارزميـات الحلّ؛ لحلّ كل المشـكلات في مجموعـة بيانـات معيّنـة، ثم تُظهر متوسـطَ التباطُؤ الـذي تحققـه كل خوارزميـة حلّ على كل المشـكلات في مجموعة البيانـات، وتَقبل الدالـة كذلك المُعامِل المنطقي use_brute لتحديد إمكانية استخدام خوارزمية الحلّ بالقوة المُفرطة أم لا:

```
from collections import defaultdict
import numpy

def compare(problems,use_brute):
    # comparison on Dataset 1
    # maps each solver to a list of all tardiness values it achieves for the problems in the given dataset
    results = defaultdict(list)
    for problem in problems: # for each problem in this datset

        #uses each of the solvers on this problem
        if use_brute == True:
            results['brute-force'].append(brute_force_solver(problem)
['tardiness'])
        results['greedy-deadline'].append(greedy_solver(problem,deadline_
heuristic)['tardiness'])
        results['greedy-weighted_deadline'].append(greedy_
solver(problem,weighted_deadline_heuristic)['tardiness'])
        results['ls-random-wdeadline'].append(local_search_solver(problem,
weighted_deadline_heuristic, random_swap, 1000)['tardiness'])
        results['ls-random-deadline'].append(local_search_solver(problem,
deadline_heuristic, random_swap, 1000)['tardiness'])
        results['ls-adjacent-wdeadline'].append(local_search_solver(problem,
weighted_deadline_heuristic, adjacent_swap, 1000)['tardiness'])
        results['ls-adjacent-deadline'].append(local_search_solver(problem,
deadline_heuristic, adjacent_swap, 1000)['tardiness'])

    for solver in results: # for each solver
        # prints the solver's mean tardiness values
        print(solver,numpy.mean(results[solver]))
```

يُمكن الآن استخدام دالة ( )compare مع مجموعتي البيانات problems_7 و problems_30 كلتيهما:

```
compare(problems_7,True)
```

```
brute-force 211.49
greedy-deadline 308.14
greedy-weighted_deadline 255.61
ls-random-wdeadline 212.35
ls-random-deadline 212.43
ls-adjacent-wdeadline 220.62
ls-adjacent-deadline 224.36
```

```
compare(problems_30,False)
```

```
greedy-deadline 10126.18
greedy-weighted_deadline 8527.61
ls-random-wdeadline 6647.73
ls-random-deadline 6650.99
ls-adjacent-wdeadline 6666.47
ls-adjacent-deadline 6664.67
```

وزارة التعليم
Ministry of Education
2025 - 1447

# تمرينات

1 صِف استراتيجيتين مُختلفتين (مبادلة، انعكاس، تحويل، إلخ) لأسلوب البحث المحلي لِحلّ مشكلة التباطؤ الموزون للآلة الواحدة.

2 كم عدد الجداول المُمكنة (الحلول) لنسخة مشكلة التباطؤ الموزون للآلة الواحدة والتي تشتمل على تسع مهام؟



وزارة التعليم
Ministry of Education
2025 - 1447

**3** أنشئ خوارزمية حلّ بالقوة المُفرطة لمشكلة التباطؤ الموزون للآلة الواحدة، من خلال إكمال المقطع البرمجي التالي بحيث تستخدِم الدالةُ القوةَ المُفرطة لإيجاد تبديل الجدولة الأمثل.

```
def brute_force_solver(problem):
    # gets the information for this problem
    durations, weights, deadlines=problem['durations'], problem['weights'],
problem['deadlines']

    job_num = len(_______________) # number of jobs
    # generates all possible schedules

    all_schedules = itertools._______________(range(job_num))
    # initializes the best solution and its total weighted tardiness

    best_schedule = ____________________ # initialized to None
    # 'inf' stands for 'infnity'. Python will evaluate all numbers as smaller than this value.

    best_tardiness = float('__________________')
    # stores the finish time of each job in the best schedule

    best_finish_times=__________________ # initalized to None

    for schedule in all_schedules: # for every possible schedule
        #evalute the schedule
        tardiness,finish_times=compute_schedule_tardiness(problem, schedule)
        if tardiness<best_tardiness: # this schedule is better than the best so far

            best_tardiness=____________________

            best_schedule=______________

            best_finish_times=________________

    # return the results as a dictionary
    return  {'schedule':best_schedule,
             'tardiness':best_tardiness,
             'finish_times':best_finish_times}
```

4 أنشئ خوارزمية حلّ البحث المحلّي لمشكلة التباطؤ الموزون للآلة الواحدة، من خلال إكمال المقطع البرمجي التالي بحيث تستخدِم الدالةُ البحث المحلّي لإيجاد تبديل الجدولة الأمثل.

```
def local_search_solver(problem, greedy_heuristic, swap_selector, max_
iterations):
    # gets the information for this problem
    durations, weights, deadlines=problem['durations'], problem['weights'],
problem['deadlines']

    job_num = len(_______________)# gets the number of jobs
    # uses the greedy solver to get a first schedule.
    # this schedule will be then iteratively refined through local search

    greedy_sol = ________________(problem, greedy_heuristic) # remembers the best
schedule so far
    best_schedule, best_tardiness, best_finish_times=greedy_
sol['schedule'],greedy_sol['tardiness'],greedy_sol['finish_times']

    # local search

    for i in range(______________): # for each of the given iterations
        # chooses which two positions to swap

        pos1,pos2=_______________(best_schedule)

        new_schedule = best_schedule.________________()# creates a copy of the
schedule
        # swaps jobs at positions pos1 and pos2
        new_schedule[pos1], new_schedule[pos2] = best_schedule[pos2], best_
schedule[pos1]
        # computes the new tardiness after the swap
        new_tardiness, new_finish_times = compute_schedule_tardiness(problem,
new_schedule)
        # if the new schedule is better than the best one so far
        if new_tardiness < best_tardiness:
            # the new_schedule becomes the best one

            best_schedule = __________________

            best_tardiness = _________________

            best_finish_times=________________

    # returns the best solution
    return {'schedule':best_schedule,
            'tardiness':best_tardiness,
            'finish_times':best_finish_times}
```


وزارة التعليم
Ministry of Education
2025 - 1447

**5** صِف طريقة عمل البحث المحلّي.

**6** اكتب ملاحظاتك عن نتائج خوارزميات الحلّ الجشعة مقارنة بخوارزميات حلّ البحث المحلّي في مشكلة تشتمل على ثلاثين مُهمَّة. من وجهة نظرك، لماذا لم تُستخدم خوارزمية حلّ القوة المُفرطة في هذه المشكلة المكوّنة من ثلاثين مُهِمَّة؟

وزارة التعليم
Ministry of Education
2025 - 1447

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الثالث

# مشكلة تحسين المسار

## البرمجة الرياضية في مشكلات التحسين
## Mathematical Programming in Optimization Problems

في الدرسين السابقين تم توضيح كيفية استخدام الخوارزميات الاستدلالية لحلّ أنواع مُختلفة من مشكلات التحسين، وبالرغم من أن الاستدلالات بإمكانها أن تكون سريعة جدًا وتُنتج في العادة حلولًا جيدة، إلا أنها لا تضمن دائما إيجاد الحلّ الأمثل، وقد لا تكون مناسبة لكل أنواع المشكلات، وفي هذا الدرس ستُركِّز على أسلوب تحسين مُختلف وهو **البرمجة الرياضية** (Mathematical Programming).

**البرمجة الرياضية (Mathematical Programming):**
هي تقنية تُستخدم لحلّ مشكلات التحسين عن طريق صياغتها على هيئة نماذج رياضية.

يُمكن للبرمجة الرياضية أن تحلّ العديد من مشكلات التحسين مثل: تخصيص الموارد، وتخطيط الإنتاج، والخدمات اللوجستية والجدولة، وتتميز هذه التقنية بأنها تُوفّر حلًّا مثاليًا مضمونًا ويُمكنها التعامل مع المشكلات المعقدة ذات القيود المتعددة.

يبدأ حلّ البرمجة الرياضية بصياغة مشكلة التحسين المُعطاة على شكل نموذج رياضي باستخدام المتغيِّرات، حيث تُمثِّل هذه المتغيِّرات القيم التي يجب تحسينها، ثم يتم استخدامها لتحديد الدالة الموضوعية والقيود، وهما يصفان المشكلة معًا ويُمكِّنان من استخدام خوارزميات البرمجة الرياضية.

تستخدِم البرمجة الرياضية **متغيِّرات القرار** (Decision Variables) التي تساعد مُتَّخِذ القرار في إيجاد الحل المناسب عن طريق ضبطها والتحكم فيها، كما يمكنها أن تستخدِم **متغيِّرات الحالة** (State Variables) التي لا يتحكم فيها مُتَّخِذ القرار وتفرضها البيئة الخارجية، وبالتالي لا يمكن ضبط متغيِّرات الحالة. تُوفِّر القوائم التالية أمثلة على متغيِّرات القرار ومتغيِّرات الحالة لبعض مشكلات التحسين الشائعة:

**جدول 5.2: أمثلة على متغيِّرات القرار ومتغيِّرات الحالة**

| | متغيِّرات القرار | متغيِّرات الحالة |
|---|---|---|
| تخطيط الإنتاج | الكمية التي يجب إنتاجها من كل مُنتَج. | تَوفّر المواد الخام، وسعة آلات الإنتاج، وتوفّر العمالة المطلوبة للإنتاج. |
| نقل الموارد | عدد السلع التي يجب نقلها من مكان لآخر. | المسافة بين الأماكن التي يجب زيارتها وسعة المركبات. |
| جدولة المهام | ترتيب كل مُهِمَّة والمدة الزمنية اللازمة لإجرائها. | تَوفّر العمّال والآلات، والمواعيد النهائية، ووزن أهمية كل مُهِمَّة. |
| توزيع الموظفين حسب المهام | تكليف العمّال وجدولتهم للقيام بمهام مُختلفة في أوقات مُختلفة. | مهارات كل عامل وتفضيلاته، وجاهزيّته، والمهارات المطلوبة منه لإنجاز كل مُهِمَّة. |


وزارة التعليم
Ministry of Education
2025 - 1447

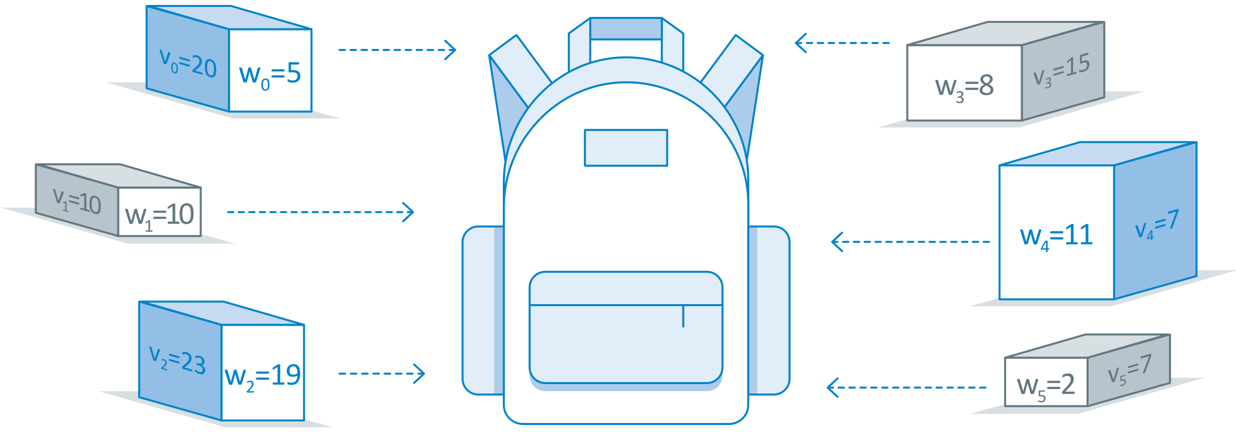
تتم صياغة الدالة الموضوعية **كتعبير رياضي** (Mathematical Expression) لتحسينها (بزيادتها أو تقليلها) بناءً على المتغيِّرات المناسبة، وتُمثِّل هذه الدالة الهدف من مشكلة التحسين مثل: زيادة الربح أو تقليل التكاليف، وتُحدَّد في العادة بناءً على متغيِّرات القرار، كما تُحدَّد أحيانًا بناءً على متغيِّرات الحالة، وبالمثل يُمكن صياغة القيود باستخدام المتغيِّرات والمتباينات الرياضية.

توجد عدة أنواع من البرمجة الرياضية، مثل: **البرمجة الخطية** (Linear Programming - LP)، و**البرمجة الرباعية** (Quadratic Programming - QP) و**برمجة الأعداد الصحيحة المختلطة** (Mixed Integer Programming - MIP). يركِّز هذا الدرس على برمجة الأعداد الصحيحة المختلطة المُستخدَمة في المشكلات التي تتقيّد فيها متغيِّرات القرار بالأعداد الصحيحة مثل: مشكلات الجدولة أو اختيار الطريق.

## مشكلة حقيبة الظهر The Knapsack Problem

مشكلة حقيبة الظهر 1/0 هي مثال بسيط على استخدام برمجة الأعداد الصحيحة المختلطة لصياغة الدالة الموضوعية والقيود، وتُعرَّف المشكلة على النحو التالي: لديك حقيبة ظهر بسعة قصوى تبلغ $C$ وحدة، ومجموعة من العناصر $I$، بحيث يكون لكل عنصر $i$ متغيِّران من متغيِّرات الحالة هما وزن العنصر $w_i$ وقيمته $v_i$، والمطلوب هو تعبئة الحقيبة بمجموعة العناصر ذات أقصى قيمة ممكنة في حدود سعة الحقيبة. يُستخدم متغيِّر القرار $x_i$ لتتبع تجميعات العناصر التي ستُعبَّأ في حقيبة الظهر، حيث تكون $x_i = 1$ إذا تم اختيار العنصر $i$ للإضافة للحقيبة، بينما تكون $x_i = 0$ خلاف ذلك، ويتمثّل الهدف في انتقاء مجموعة فرعية من العناصر من $I$ بحيث تشمل:

- **القيد** (Constraint): مجموع أوزان العناصر المنتقاة بها لا يزيد عن السعة القصوى $C$.
- **الدالة الموضوعية** (Objective Function): مجموع قيم العناصر المنتقاة بها هي أقصى قيمة مُمكنة.




شكل 5.6: مشكلة حقيبة الظهر

يوضِّح الشكل 5.6 مثالًا على مسألة حقيبة ظهر مُكوَّنة من ستة عناصر بأوزان وقِيم محدَّدة، وحقيبة ظهر بسعة قصوى تساوي أربعين وحدة. يقوم المقطع البرمجي التالي بتثبيت مكتبة البايثون المفتوحة المصدر mip الخاصة ببرمجة الأعداد الصحيحة المختلطة لحلّ نسخة مشكلة حقيبة الظهر 1/0، ويستورد الوحدات الضرورية:

```
!pip install mip # install the mip library
```

```
# imports useful tools from the mip library
from mip import Model, xsum, maximize, BINARY
values = [20, 10, 23, 15, 7, 7] # values of available items
weights = [5, 10, 19, 8, 11, 2] # weights of available items
```


وزارة التعليم
Ministry of Education
2025 - 1447

```
C = 40 # knapsack capacity

I = range(len(values)) # creates an index for each item: 0,1,2,3,...

solver = Model("knapsack") # creates a knapsack solver
solver.verbose = 0 # setting this to 1 will print more information on the progress of the solver

x = [] # represents the binary decision variables for each item.

# for each items creates and appends a binary decision variable
for i in I:
    x.append(solver.add_var(var_type = BINARY))

# creates the objective function
solver.objective = maximize(xsum(values[i] * x[i] for i in I))

# adds the capacity constraint to the solver
solver += xsum(weights[i] * x[i] for i in I) <= C

# solves the problem
solver.optimize()
```

```
<OptimizationStatus.OPTIMAL: 0>
```

يُنشئ المقطع البرمجي القائمة *x* لتخزين متغيِّرات القرار الثنائية للعناصر، وتُوفِّر المكتبة mip في البايثون ما يلي:

- أداة add_var(var_type=BINARY) لإنشاء المتغيِّرات الثنائية وإضافتها إلى خوارزمية الحلّ.
- أداة ( )maximize لمشكلات التحسين التي تحتاج لزيادة دالة موضوعية، أما مشكلات التحسين التي تتطلب تصغير الدالة الموضوعية، فتستخدِم الأداة ( )minimize.
- أداة ( )xsum لإنشاء التعبيرات الرياضية التي تتضمن **المجاميع** (sums)، وفي المثال السابق تم استخدام هذه الأداة لحساب مجموع الوزن الإجمالي للعناصر في إنشاء قيد السّعة وحلّه.
- أداة ( )optimize لإيجاد حلّ يحسِّن الدالة الموضوعية في ظل الالتزام بالقيود، وتَستخدم الأداة برمجةَ الأعداد الصحيحة المختلطة للنظر بكفاءة في توليفات القيم المُختلفة لمتغيِّرات القرار ولإيجاد التوليفة التي تُحسِّن الهدف.
- المُعامِل =+ لإضافة قيود إضافية إلى خوارزمية الحلّ الموجودة.

في المقطع البرمجي أدناه تحتوي القائمة x على متغيِّر ثنائي واحد لكل عنصر، وبعد حساب الحلّ سيكون كل متغيِّر مساويًا للواحد إذا أُدرج العنصر في الحلّ، وسيُساوي صفرًا بخلاف ذلك. تَستخدم المكتبة mip بناء الجملة x[i].x لإظهار القيمة الثنائية للعنصر ذي الفهرس i، وتَحسب خوارزمية الحلّ متغيِّر القرار x، ثم تجد القيمة الإجمالية والوزن الإجمالي للعناصر المنتقاة عن طريق التكرار على متغيِّر القرار x، وتَجمع الأوزان والقيم لكل عنصر منتقى *i*، استنادًا إلى x[i]، وتَعرضها كما هو موضَّح في المقطع البرمجي التالي:

```
total_weight = 0 # stores the total weight of the items in the solution
total_value = 0 # stores the total value of the items in the solution
```


وزارة التعليم
Ministry of Education
2025 - 1447

```
for i in I: # for each item
    if x[i].x == 1: # if the item was selected
        print('item', i, 'was selected')
        # updates the total weight and value of the solution
        total_weight += weights[i]
        total_value += values[i]

print('total weight', total_weight)
print('total value', total_value)
```

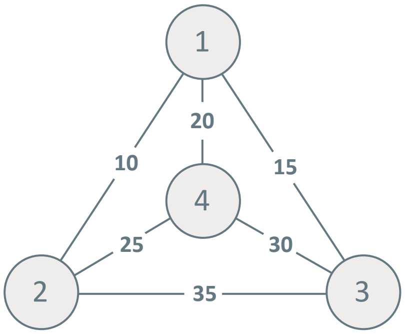
```
item 0 was selected
item 2 was selected
item 3 was selected
item 5 was selected
total weight 34
total value 65
```

## مشكلة البائع المُتجوّل
## Traveling Salesman Problem

**مشكلة البائع المُتجوّل** (Traveling Salesman Problem – TSP) من المشكلات الأخرى التي يُمكن حلّها ببرمجة الأعداد الصحيحة المختلطة، وهي مشكلة مألوفة تُعنى بتحديد أقصر مسار يمكن أن يسلكه بائع متجوِّل لزيارة مدنٍ معيّنة مرة واحدة، دون أن يكرّر زيارة أيٍّ منها، ثمّ يعودَ للمدينة الأصلية، ويصوِّر الشكل 5.7 نسخةً من هذه المشكلة.

الأمثلة الواردة في مُخطَّط مشكلة البائع المُتجوّل متصلة اتصالًا تامًّا؛ فهناك حافة تصل كل زوج من العُقد بالآخر.

تُمثِّل كل دائرة (عقدة) مدينة أو موقعًا يجب زيارته، وهناك حافة تربط بين موقعين إذا كان من الممكن السفر بينهما، ويُمثِّل الرقم الموجود على الحافة التكلفة (المسافة) بين الموقعين. في هذا المثال، تم ترقيم المواقع وفقًا لترتيبها في الحلّ الأمثل للمشكلة، وتكون التكلفة الإجمالية للطريق 1←2←4←3←1 تساوي 80 = 15+30+25+10، وهو أقصر طريق ممكن لزيارة كل مدينة مرة واحدة فقط والعودة إلى نقطة البداية. توجد تطبيقات عملية لمشكلة البائع المُتجوّل في الخدمات اللوجستية، والنقل، وإدارة الإمدادات والاتصالات، فهي تنتمي إلى عائلة أوسع من مشكلات تحديد الطريق التي تشمل أيضًا مشكلات شهيرة أخرى موضحة فيما يلي:



شكل 5.7: نسخة على مشكلة البائع المُتجوّل

- تتضمن **مشكلة تحديد مسار المركبات** (Vehicle Routing Problem) إيجاد الطُرق المُثلى لأسطول من المركبات لتوصيل السلع أو الخدمات لمجموعة من العملاء في ظل تقليل المسافة الإجمالية المقطوعة إلى الحد الأدنى، وتشمل تطبيقاتها الخدمات اللوجستية وخدمات التوصيل وجمع النفايات.
- تتضمن **مشكلة الاستلام والتسليم** (Pickup and Delivery Problem) إيجاد الطُرق المُثلى للمركبات لكي تستلم (تُحمِّل أو تُركِب) وتسلِّم (تُوصل) البضائع أو الأشخاص إلى مواقع مُختلفة، وتشمل تطبيقاتها خدمات سيارات الأجرة، والخدمات الطبية الطارئة، وخدمات النقل الجماعي.
- تتضمن **مشكلة جدولة مواعيد القطارات** (Train Timetabling Problem) إيجاد جداول زمنية مثالية للقطارات في شبكة سكك الحديد في ظل تقليل نسبة التأخير إلى الحد الأدنى وضمان الاستخدام الفعّال للموارد، وتشمل تطبيقاتها النقل بالسكك الحديدية والجدولة.

يُمكن استخدام المقطع البرمجي التالي لإنشاء نسخة من مشكلة البائع المُتجوّل، وتَقبل الدالة عدد المواقع المراد زيارتها، ونطاق المسافة يُمثِّل الفَرق بين المسافة الأقصر والمسافة الأطول بين موقعين، ثم تُظهر:

- مصفوفة المسافة التي تشمل المسافة المُسندة بين كل زوج ممكن من المواقع.
- مجموعة عناوين المواقع العددية (عنوان لكل موقع).
- الموقع الذي يكون بمثابة بداية الطريق ونهايته، ويُشار إليه باسم موقع startstop (الانطلاق والتوقف).

```python
import random
import numpy
from itertools import combinations

def create_problem_instance(num_locations, distance_range):
    # initializes the distance matrix to be full of zeros
    dist_matrix = numpy.zeros((num_locations, num_locations))
    # creates location ids: 0,1,2,3,4,...
    location_ids = set(range(num_locations))
    # creates all possible location pairs
    location_pairs = combinations(location_ids, 2)
    for i,j in location_pairs: # for each pair
        distance = random.randint(*distance_range) # samples a distance within range
        # the distance from i to j is the same as the distance from j to i
        dist_matrix[j,i] = distance
        dist_matrix[i,j] = distance

    # returns the distance matrix, location ids and the startstop vertix
    return dist_matrix, location_ids, random.randint(0, num_locations - 1)
```

يستخدِم المقطع البرمجي التالي الدالة الواردة سابقًا لإنشاء نسخة من مشكلة البائع المُتجوّل، بحيث يتضمن 8 مواقع، ومسافات ثنائية تتراوح بين 5 و20:

```python
dist_matrix, location_ids, startstop = create_problem_instance(8, (5, 20))
print(dist_matrix)
print(startstop)
```

```
[[ 0. 19. 17. 15. 18. 17.  7. 15.]
 [19.  0. 15. 18. 11.  6. 20.  5.]
 [17. 15.  0. 17. 15.  7.  5. 11.]
 [15. 18. 17.  0. 19.  7.  7. 16.]
 [18. 11. 15. 19.  0. 17. 20. 17.]
 [17.  6.  7.  7. 17.  0. 15. 14.]
 [ 7. 20.  5.  7. 20. 15.  0. 14.]
 [15.  5. 11. 16. 17. 14. 14.  0.]]
3
```

لاحظ أن الخط القُطري يُمثِّل المسافات من العُقد إلى نفسها (dist_ matrix[i،i])، وبالتالي فإن المسافات تساوي أصفارًا.

## إنشاء خوارزمية حلّ القوة المُفرطة لمشكلة البائع المُتجوّل
## Creating a Brute-Force Solver for the Traveling Salesman Problem

تستخدِم الدالة التالية خوارزمية حلّ القوة المُفرطة لتعداد جميع الطُرق المُمكنة (التباديل) وإظهار أقصر مسار، وتَقبل هذه الدالة مصفوفة المسافة وموقع الانطلاق والتوقف الذي تُظهره الدالة ( )create_problem_instance. لاحظ أن الحل الممكن لنسخة **مشكلة البائع المُتجوّل** (TSP) هي تبديل مدن، يبدأ من مدينة startstop (الانطلاق والتوقف) ثم ينتهي إليها.

```python
from itertools import permutations

def brute_force_solver(dist_matrix, location_ids, startstop):
    # excludes the starstop location
    location_ids = location_ids - {startstop}
    # generate all possible routes (location permutations)
    all_routes = permutations(location_ids)
    best_distance = float('inf') # initializes to the highest possible number
    best_route = None # best route so far, initialized to None

    for route in all_routes: # for each route
        distance = 0 # total distance in this route
        curr_loc = startstop # current location

        for next_loc in route:
            distance += dist_matrix[curr_loc,next_loc] # adds the distance of this step
            curr_loc = next_loc # goes to the next location
        distance += dist_matrix[curr_loc,startstop] # goes to the starstop location
        if distance < best_distance: # if this route has lower distance than the best route
            best_distance = distance
            best_route = route

    # adds the startstop location at the beginning and end of the best route and returns
    return [startstop] + list(best_route) + [startstop], best_distance
```

تستخدِم خوارزمية حلّ القوة المُفرطة أداة ( )permutations لإنشاء كل الطُرق المُمكنة. لاحظ أن موقع startstop (الانطلاق والتوقف) يُستبعد من التباديل؛ لأنه يجب أن يظهر دائمًا في بداية كلّ طريق ونهايته، فعلى سبيل المثال، إذا كانت لديك أربعة مواقع 0، و1، و2، و3، وكان الموقع 0 هو موقع startstop (الانطلاق والتوقف)، ستكون قائمة التباديل المُمكنة كما يلي:

```python
for route in permutations({1,2,3}):
    print(route)
```

```
(1, 2, 3)
(1, 3, 2)
(2, 1, 3)
(2, 3, 1)
(3, 1, 2)
(3, 2, 1)
```

تَحسب خوارزمية حلّ القوة المُفرطة المسافة الإجمالية لكل طريق، وتُظهر في النهاية الطريق ذا المسافة الأقصر. يُطبِّق المقطع البرمجي التالي خوارزمية الحلّ على نسخة مشكلة البائع المُتجوّل التي تم إنشاؤها سابقًا:

```
brute_force_solver(dist_matrix, location_ids, startstop)
```

```
([3, 5, 2, 7, 1, 4, 0, 6, 3], 73.0)
```

على غرار خوارزميات حلّ القوة المُفرطة التي تم توضيحها في الدروس السابقة، لا تُطبَّق هذه الخوارزمية إلا على نُسخ مشكلة البائع المُتجوّل الصغيرة؛ لأن عدد الطُرق المُمكنة يتزايد أضعافًا مضاعفة كلما زاد العدد $N$، ويساوي $(N-1)!$، وعلى سبيل المثال، عندما يكون $N = 15$، فإن عدد الطُرق المُمكنة يساوي $14! = 87,178,291,200$.

## استخدام برمجة الأعداد الصحيحة المختلطة لحلّ مشكلة البائع المُتجوّل
## Using MIP to Solve the Traveling Salesman Problem

لاستخدام **برمجة الأعداد الصحيحة المختلطة** (MIP) لحلّ **مشكلة البائع المُتجوّل** (TSP)، يجب إنشاء صيغة رياضية تُغطي كلًا من الدالة الموضوعية وقيود مشكلة البائع المُتجوّل.

تتطلب الصيغة متغيِّر قرار ثنائي $x_{ij}$ لكل انتقال محتمل $i \leftarrow j$ من موقع $i$ إلى موقع آخر $j$، وإذا كانت المشكلة بها عدد $N$ من المواقع، فإن عدد الانتقالات المُمكنة يساوي $N\times(N-1)$. إذا كانت $x_{ij}$ تساوي 1، فإن الحلّ يتضمن الانتقال من الموقع $i$ إلى الموقع $j$، وخلاف ذلك إذا كانت $x_{ij}$ تساوي 0، فلن يُدرج هذا الانتقال في الحلّ.

يُمكن الوصول بسهولة إلى العناصر في مصفوفة numpy ثنائية الأبعاد عبر الصيغة البرمجية [i,j]، فعلى سبيل المثال:

```
arr = numpy.full((4,4), 0) # creates a 4x4 array full of zeros

print(arr)

arr[0, 0] = 1
arr[3, 3] = 1

print()
print(arr)
```

```
[[0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]]

[[1 0 0 0]
 [0 0 0 0]
 [0 0 0 0]
 [0 0 0 1]]
```

يستخدِم المقطع البرمجي الأداة ( )product من المكتبة itertools لحساب جميع انتقالات المواقع المحتملة، فعلى سبيل المثال:

```
ids = {0, 1, 2}
for i, j in list(product(ids, ids)):
    print(i, j)
```

```
0 0
0 1
0 2
1 0
1 1
1 2
2 0
2 1
2 2
```

يستخدِم المقطع البرمجي التالي مكتبة البايثون mip لإنشاء خوارزمية حلّ برمجة الأعداد الصحيحة المختلطة، ثم يضيف متغيِّر قرار ثُنائي لكل انتقال ممكن في نسخة مشكلة البائع المُتجوّل التي تم إنشاؤها سابقًا:

```
from itertools import product # used to generate all possible transition
from mip import BINARY
from mip import Model,INTEGER

solver = Model() # creates a solver
solver.verbose = 0 # setting this to 1 will print info on the progress of the solver

# 'product' creates every transition from every location to every other location
transitions = list(product(location_ids, location_ids))

N = len(location_ids) # number of locations

# creates a square numpy array full of 'None' values
x = numpy.full((N, N), None)

# adds binary variables indicating if transition (i->j) is included in the route
for i, j in transitions:
    x[i, j] = solver.add_var(var_type = BINARY)
```

يستخدِم المقطع البرمجي السابق أداة numpy.full( ) لإنشاء مصفوفة numpy بحجم NxN لتخزين المتغيِّرات الثنائية x.

بعد إضافة متغيِّرات القرار x، يُمكن استخدام المقطع البرمجي التالي لصياغة وحساب الدالة الموضوعية لمشكلة البائع المُتجوّل، حيث تقوم الدالة بالتكرار على كل انتقال ممكن $i \rightarrow j$ وتُضرب مسافتها dist_matrix[i,j] مع متغيِّر قرارها x[i,j]، وإذا تم إدراج الانتقال في الحلّ سيؤخذ x[i,j]=1 و dist_matrix[i,j] بعين الاعتبار، وبخلاف ذلك ستُضرب dist_matrix[i,j] في صفر ليتم تجاهُلها:

```
# the minimize tool is used then the objective function has to be minimized
from mip import xsum, minimize

# objective function: minimizes the distance
solver.objective = minimize(xsum(dist_matrix[i,j]*x[i][j] for i,j in
transitions))
```

تهدف الخطوة التالية إلى التأكد بأن الخوارزمية تُظهر الحلول التي تضمن زيارة كل المواقع لمرَّة واحدة فقط، باستثناء موقع startstop (الانطلاق والتوقف) حسب ما تتطلبه مشكلة البائع المُتجوّل، وزيارة كل موقع مرة واحدة تعني أن الطريق الصحيح يُمكن أن:

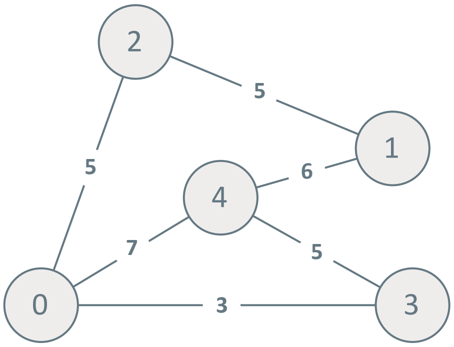
- يصل إلى كل موقع مرة واحدة فقط.
- يغادر من كل موقع مرة واحدة فقط.

ويُمكن إضافة قيود الوصول والمغادرة هذه بسهولة كما يلي:

```
# for each location id
for i in location_ids:
    solver += xsum(x[i,j] for j in location_ids - {i}) == 1 # exactly 1 arrival
    solver += xsum(x[j,i] for j in location_ids - {i}) == 1 # exactly 1 departure
```

وزارة التعليم
Ministry of Education
2025 - 1447

تشمل الصيغة الكاملة لمشكلة البائع المُتجوّل نوعًا إضافيًا آخرًا من القيود لضمان حساب الطُّرق المتصلة، ففي نسخة مشكلة البائع المُتجوّل الواردة في الشكل 5.8 يُفترض أن الموقع 0 هو موقع الانطلاق والتوقف.



شكل 5.8: نسخة مشكلة البائع المُتجوّل

في هذا المثال، أقصر طريق ممكن هو 0 ← 3 ← 4 ← 1 ← 2، بمسافة سفر إجمالية قدرها 24، ولكن عند عدم وجود قيد اتصال سيكون هناك حلّ صحيح آخر يشمل طريقين غير متصلين هما: 0 ← 3 ← 4 ← 0 و 1 ← 2 ← 1، وهذا الحلّ المتمثل في وجود طريقين يمتثل لقيود الوصول والمغادرة التي تم تعريفها في المقطع البرمجي السابق؛ لأن كل موقع يدخل له ويخرج منه مرة واحدة فقط، ولكن هذا الحلّ غير مقبول لمشكلة البائع المُتجوّل.

يُمكن فرض حلّ يشمل طريقًا واحدًا متصلًا بإضافة متغيِّر القرار $y_i$ لكل موقع i، وستحافظ هذه المتغيِّرات على ترتيب زيارة كل موقع في الحلّ.

```
# adds a decision variable for each location
y = [solver.add_var(var_type = INTEGER) for i in location_ids]
```

على سبيل المثال، إذا كان الحلّ هو: 0 ← 3 ← 4 ← 1 ← 2 ← 0، فستكون قيم y كما يلي: $y_3$=0، $y_4$=1، $y_1$=2، $y_2$=3، والموقع 0 هو موقع الانطلاق والتوقف، ولذلك لا تؤخذ قيمة y الخاصة به بعين الاعتبار.

يُمكن استخدام متغيِّرات القرار الجديدة هذه لضمان الاتصال من خلال إضافة قيد جديد لكل انتقال ***i*** ← ***j*** لا يشمل موقع startstop (الانطلاق والتوقف).

```
# adds a connectivity constraint for every transition that does not include the startstop
for (i, j) in product(location_ids - {startstop}, location_ids - {startstop}):
    # ignores transitions from a location to itself
    if i != j:
        solver += y[j] - y[i] >= (N+1) * x[i, j] - N
```

إذا كانت $x_{ij}=1$ لانتقال ***i*** ← ***j*** وتم إدراج هذا الانتقال في الحلّ، فإن المتباينة الواردة في الأعلى تصبح y[j] >= y[i] + 1، ومعنى ذلك أن المواقع التي ستُزار لاحقًا لا بد أن تكون قيمة y الخاصة بها أعلى، بالإضافة إلى قيود الوصول والمغادرة، وسيكون الطريق الذي لا يشمل موقع الانطلاق والتوقف صحيحًا فقط إذا:

- بدأ وانتهى بالموقع نفسه؛ لضمان أن يكون لكل موقع وصولٌ واحدٌ ومغادرة واحدة فقط.
- خُصصت قيم y أعلى لكل المواقع التي ستُزارُ لاحقًا؛ لأن y[j] يجب أن تكون أكبر من y[i] لكل الانتقالات التي تم إدراجها في الطريق، ويؤدي هذا أيضًا إلى تجنب إضافة الحافة نفسها من اتجاه مُختلف، على سبيل المثال: ***i*** ← ***j*** و ***j*** ← ***i***

ولكن إذا كان الموقع يمثل بداية الطريق ونهايته، فلا بدّ أن تكون قيمة y الخاصة به هي أكبر وأصغر من قيم كل المواقع الباقية في الطريق، ونظرًا لاستحالة هذا الأمر، فستُؤدي إضافة قيد الاتصال إلى استبعاد أية حلول بها طُرق لا تشمل موقع الانطلاق والتوقف.


وزارة التعليم
Ministry of Education
2025 - 1447

على سبيل المثال، فكِّر في الطريق 1 ← 2 ← 1 الوارد في الحلّ المُكوَّن من طريقين لنسخة مشكلة البائع المُتجوّل الموضحة في الشكل السابق، حيث يتطلب قيد الاتصال أن تكون $y_2 \geq y_1 + 1$ وأن تكون $y_1 \geq y_2 + 1$، وهذا مستحيل، فلذلك سيتم استبعاد الحلّ.

في المقابل، يتطلب الحلّ الصحيح 0 ← 3 ← 4 ← 1 ← 2 ← 0 أن تكون $y_4 \geq y_3 + 1$ وأن تكون $y_1 \geq y_4 + 1$، وأن تكون $y_2 \geq y_1 + 1$، ويُمكن تحقيق ذلك بضبط قيم y كما يأتي: $y_3=0$ و $y_4=1$ و $y_1=2$ و $y_2=3$، ولا تنطبق قيود الاتصال على الانتقالات التي تشمل موقع startstop (الانطلاق والتوقف).

تَجمع الدالة التالية كل الأشياء معًا لإنشاء خوارزمية حلّ برمجة الأعداد الصحيحة المختلطة لمشكلة البائع المُتجوّل:

```
from itertools import product
from mip import BINARY,INTEGER
from mip import Model
from mip import xsum, minimize

def MIP_solver(dist_matrix, location_ids, startstop):
    solver = Model()# creates a solver
    solver.verbose = 0 # setting this to 1 will print info on the progress of the solver
    # creates every transition from every location to every other location
    transitions = list(product(location_ids,location_ids))
    N = len(location_ids) # number of locations
    # create an empty square matrix full of 'None' values
    x = numpy.full((N, N), None)
    # adds binary decision variables indicating if transition (i->j) is included in the route
    for i, j in transitions:
        x[i, j]=solver.add_var(var_type = BINARY)
    # objective function: minimizes the distance
    solver.objective = minimize(xsum(dist_matrix[i,j]*x[i][j] for i,j in transitions))
    # Arrive/Depart Constraints
    for i in location_ids:
        solver += xsum(x[i,j] for j in location_ids - {i}) == 1 # exactly 1 arrival
        solver += xsum(x[j,i] for j in location_ids - {i}) == 1 # exactly 1 departure
    # adds a binary decision variable for each location
    y = [solver.add_var(var_type=INTEGER) for i in location_ids]
    # adds connectivity constraints for transitions that do not include the startstop
    for (i, j) in product(location_ids - {startstop}, location_ids - {startstop}):
        if i != j: # ignores transitions from a location to itself
            solver += y[j] - y[i] >=(N+1)*x[i,j] - N
    solver.optimize() #solves the problem
    # prints the solution
    if solver.num_solutions: # if a solution was found
        best_route = [startstop] # stores the best route
        curr_loc = startstop # the currently visited location
        while True:
            for next_loc in location_ids:# for every possible next location
                if x[curr_loc,next_loc].x == 1: # if x value for the curr_loc->next_loc transition is 1
                    best_route.append(next_loc) # appends the next location to the route
                    curr_loc=next_loc # visits the next location
                    break
            if next_loc == startstop: # exits if route returns to the startstop
                break
    return best_route, solver.objective_value # returns the route and its total distance
```

وزارة التعليم
Ministry of Education
2025 - 1447

يولِّد المقطع البرمجي التالي 100 نسخة من مشكلة البائع المُتجوّل تشمل 8 مواقع وتتراوح المسافات فيها بين 5 و20، كما أنه يستخدِم خوارزمية حلّ القوة المُفرطة، وخوارزمية حلّ برمجة الأعداد الصحيحة المختلطة لحلّ كل حالة، ويُظهِر النسبة المئوية للأسلوبين اللذين أظهرا طريقين لهما المسافة نفسها:

```
same_count = 0
for i in range(100):
    dist_matrix, location_ids, startstop=create_problem_instance(8, [5,20])
    route1, dist1 = brute_force_solver(dist_matrix, location_ids, startstop)
    route2, dist2 = MIP_solver(dist_matrix, location_ids, startstop)
    # counts how many times the two solvers produce the same total distance
    if dist1 == dist2:
        same_count += 1
print(same_count / 100)
```

```
1.0
```

تؤكد النتائج أن خوارزمية حلّ برمجة الأعداد الصحيحة المختلطة تُظهِر الحلّ الأمثل بنسبة 100% لكل نُسخ المشكلة، ويوضِّح المقطع البرمجي التالي سرعة خوارزمية حلّ برمجة الأعداد الصحيحة المختلطة من خلال استخدامها لحلّ 100 نسخة كبيرة تتضمن كلٌّ منها 20 موقعًا:

```
import time

start = time.time() # starts timer
for i in range(100):
    dist_matrix, location_ids, startstop = create_problem_instance(20, [5,20])
    route, dist = MIP_solver(dist_matrix, location_ids, startstop)

stop=time.time() # stops timer
print(stop - start) # prints the elapsed time in seconds
```

```
188.90074133872986
```

على الرغم من أن وقت التنفيذ الدقيق سيعتمد على قوة معالجة الجهاز الذي تستخدِمه لتنفيذ مفكرة جوبيتر، إلا أنه من المفترض أن يستغرق التنفيذ بضع دقائق لحساب الحلّ لجميع مجموعات البيانات المئة.

وهذا بدوره مذهل إذا تم الأخذ في الاعتبار أن عدد الطُرق المُمكنة لكل نسخة من النُسخ المئة هي: 19! = 121,645,100,000,000,000 طريقًا مُختلفًا، ومثل هذا العدد الكبير من الطُرق يفوق بكثير قدرات أسلوب القوة المُفرطة، ومع ذلك فإنه عن طريق البحث الفعّال في هذه المساحة الهائلة الخاصة بجميع الحلول المُمكنة يُمكن لخوارزمية حلّ برمجة الأعداد الصحيحة المختلطة أن تجد الطريق الأمثل بسرعة.

وعلى الرغم من مزايا البرمجة الرياضية إلا أنها تملك قيودًا خاصة أيضًا، فهي تتطلب فهمًا قويًا للنمذجة الرياضية وقد لا تكون مناسبة للمشكلات المعقدة التي يصعب فيها التعبير عن الدالة الموضوعية والقيود بواسطة الصيغ الرياضية، وعلى الرغم من أن البرمجة الرياضية أسرع بكثير من أسلوب القوة المُفرطة إلا أنها قد تظل بطيئة جدًا بالنسبة لمجموعات البيانات الكبيرة، وفي مثل هذه الحالات يقدِّم الأسلوب الاستدلالي الموضَّح في الدرسين السابقين بديلًا أكثر سرعة.

# تمرينات

1 اشرح طريقة استخدام البرمجة الرياضية لحلّ مشكلات التحسين المعقدة.

2 ما مزايا وعيوب أسلوب برمجة الأعداد الصحيحة المختلطة في حلّ مشكلات التحسين؟

وزارة التعليم
Ministry of Education
2025 - 1447

3 قم بتحليل مشكلتين من مشكلات التحسين يُمكن حلهما باستخدام البرمجة الرياضية، ثم حدِّد متغيِّرات الحالة ومتغيِّرات القرار الخاصة بهما.

4 اذكر ثلاث مشكلات تحسين مُختلفة من عائلة مشكلات تحديد المسار.


وزارة التعليم
Ministry of Education
2025 - 1447

5 أنشئ دالة خوارزمية حلّ القوة المُفرطة لمشكلة البائع المُتجوّل، من خلال إكمال المقطع البرمجي التالي بحيث تُظهر الدالة المسار الأفضل والمسافة الإجمالية المُثلى:

```
from itertools import permutations


def brute_force_solver(dist_matrix, location_ids, startstop):
    # excludes the startstop location
    location_ids = ______________ - {_______________}
    # generates all possible routes (location permutations)
    all_routes = ________________(__________________)
    best_distance = float('inf') # initializes to the highest possible number
    best_route = None # best route so far, initialized to None

    for route in all_routes: # for each route
        distance = 0 # total distance in this route
        curr_loc = ______________ # current location

        for next_loc in route:
            distance += _________[curr_loc, next_loc] # adds the distance of this step
            curr_loc = ________________ # goes the next location
        distance += ___________________[curr_loc, _______________] # goes to
back to the startstop location
        if distance < best_distance: # if this route has lower distance than the best route
            best_distance = distance
            best_route = route

    # adds the startstop location at the beginning and end of the best route and returns
    return [startstop] + list(best_route) + [startstop], best_distance
```

**6** أنشئ خوارزمية حلّ برمجة الأعداد الصحيحة المختلطة لمشكلة البائع المُتجوّل، من خلال إكمال المقطع البرمجي التالي، بحيث تنتقي متغيِّرات القرار وقيود الاتصال انتقاءً صحيحًا:

```
def MIP_solver(dist_matrix, location_ids, startstop):

    solver = ______________() # creates a solver
    solver.verbose = 0  # setting this to 1 will print info on the progress of the solver
    # creates every transition from every location to every other location

    transitions = list(_________________(location_ids, location_ids))
    N = len(location_ids) # number of locations
    # creates an empty square matrix full of 'None' values
    x = numpy.full((N, N), None)
    # adds binary decision variables indicating if transition (i->j) is included in the route
    for i, j in transitions:

        x[i, j] = solver._______________(var_type=_________________)

    # objective function: minimizes the distance

    solver.objective = _________________(xsum(dist_matrix[i, j] * x[i][j] for
i, j in transitions))

    # Arrive/Depart Constraints
    for i in location_ids:

        solver += xsum(_________________ for j in location_ids - {i}) == 1

        solver += xsum(_________________ for j in location_ids - {i}) == 1

    # Adds a binary decision variable for each location

    y = [solver._________________(var_type=_________________) for i in
location_ids]

    # Adds connectivity constraints for transitions that do not include the startstop
    for (i, j) in product(location_ids - {startstop}, location_ids -
{startstop}):
        if i != j: # ignores transitions from a location to itself
            solver += y[j] - y[i] >= (N + 1) * x[i, j] - N


    solver.________________() # solves the problem
```


وزارة التعليم
Ministry of Education
2025 - 1447

# المشروع

افترض أنك تعمل في شركة توصيل، وطلب منك مديرك أن تجد المسار الأكثر كفاءة لتوصيل الطرود إلى مواقع متعددة في المدينة.

يتمثّل الهدف في إيجاد أقصر مسار ممكن لزيارة كل موقع مرة واحدة فقط ومن ثمَّ العودة إلى موقع البدء. هذه المشكلة مثال على **مشكلة البائع المُتجوّل** (TSP).

1 ستقوم بإنشاء أمثلة متعددة على مشكلة البائع المُتجوّل تشمل مواقع عددها من 3 إلى 12، وستتراوح المسافة في كل مثال من 5 وحدات إلى 20 وحدة.

2 أنشئ دالة رسم نقاط باستخدام مكتبة matplotlib ترسم أفضل مسار تُنتجه خوارزمية الحلّ، يمكنك استخدام هذه الدالة فقط مع النسخة التي تشمل 20 موقعًا.

3 أنشئ دالة رسم نقاط باستخدام مكتبة matplotlib ترسم نقاط أداء كل من خوارزمية حلّ القوة المُفرطة وخوارزمية حلّ برمجة الأعداد الصحيحة المختلطة بالمقارنة بينهما.

4 اكتب تقريرًا موجزًا تناقش فيه النتائج التي توصلت إليها بخصوص كفاءة أداء خوارزميتي الحلّ، ومزايا وعيوب كل منهما.

وزارة التعليم
Ministry of Education
2025 - 1447

# ماذا تعلّمت

- تحديد أساليب التحسين الملائمة لحلّ المشكلات المعقدة.
- حلّ مشكلات تخصيص الموارد عن طريق تطبيق مقطع برمجي بلغة البايثون.
- حلّ مشكلات الجدولة عن طريق تطبيق مقطع برمجي بلغة البايثون.
- حلّ مشكلة حقيبة الظهر باستخدام خوارزميات التحسين المختلفة.
- حلّ مشكلة البائع المُتجوّل باستخدام خوارزميات التحسين المختلفة.

## المصطلحات الرئيسة

| | |
|---|---|
| البرمجة الرياضية | Mathematical Programming |
| برمجة الأعداد الصحيحة المختلطة | Mixed Integer Programming |
| مشكلة التحسين | Optimization Problem |
| البرمجة الرباعية | Quadratic Programing |
| مشكلة الجدولة | Scheduling Problem |
| تشكيل فريق | Team Formation |
| مشكلة البائع المُتجوّل | Traveling Salesman Problem |

| | |
|---|---|
| خوارزمية حلّ القوة المُفرطة | Brute-Force Solver |
| البرمجة القيدية | Constraint Programming |
| خوارزمية استدلالية جشعة | Greedy Heuristic Algorithm |
| خوارزمية حلّ جشعة | Greedy Solver |
| برمجة الأعداد الصحيحة | Integer Programming |
| خوارزمية حلّ مشكلة حقيبة الظهر | Knapsack Problem Solver |


وزارة التعليم
Ministry of Education
2025 - 1447

# 6. الذكاء الاصطناعي والمجتمع

سيتعرّف الطالب في هذه الوحدة على أخلاقيات الذكاء الاصطناعي وتأثيرها على تطوير أنظمته المتقدمة وتحديد توجهاتها، وسيُقيِّم مدى تأثير أنظمة الذكاء الاصطناعي واسعة النطاق على المجتمعات والبيئة، وكيفية تنظيم مثل هذه الأنظمة للاستخدام الأخلاقي المُستدام، وسيستخدِم بعد ذلك مُحاكي ويبوتس (Webots) لبرمجة طائرة مُسيَّرة على الحركة الذاتية واستكشاف منطقة ما من خلال تحليل الصور.

## أهداف التعلُّم

بنهاية هذه الوحدة سيكون الطالب قادرًا على أن:

- يُعرِّف أخلاقيات الذكاء الاصطناعي.
- يُفسِّر مدى تأثير التحيُّز والإنصاف على الاستخدام الأخلاقي لأنظمة الذكاء الاصطناعي.
- يُقيِّم كيفية حل مشكلة الشفافية وقابلية التفسير في الذكاء الاصطناعي.
- يُحلِّل كيفية تأثير أنظمة الذكاء الاصطناعي واسعة النطاق على المجتمع وكيفية وضع قوانين لتنظيمها.
- يُبرمج جهاز الطائرة المُسيَّرة على الحركة الذاتية.
- يُطوِّر نظام تحليل الصور لطائرة مُسيَّرة تُستخدم في استطلاع منطقة معينة.

## الأدوات

- ويبوتس (Webots)
- مكتبة أوبن سي في (OpenCV Library)

وزارة التعليم
Ministry of Education
2025 - 1447

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الأول

# مقدمة في أخلاقيات الذكاء الاصطناعي

## نظرة عامة على أخلاقيات الذكاء الاصطناعي

## Overview of AI Ethics

مع استمرار تقدُّم الذكاء الاصطناعي تزايدت أهمية التفكير في الآثار الأخلاقية المترتبة على استخدام هذه التقنية، ومن المهم أن يفهم المواطن في عَالَمنا الحديث الدور الهام لأخلاقيات الذكاء الاصطناعي إذا أردنا تطوير أنظمة ذكاء اصطناعي مسؤولة واستخدامها. إن أحد الأسباب الرئيسة للتأكيد على أهمية أخلاقيات الذكاء الاصطناعي هو التأثير الكبير لأنظمة الذكاء الاصطناعي على حياة الانسان. على سبيل المثال، يُمكن استخدام خوارزميات الذكاء الاصطناعي لاتخاذ قرارات التوظيف والعلاج الطبي، وإذا كانت هذه الخوارزميات مُتحيِّزة أو تمييزية، فقد تُؤدي إلى نتائج غير عادلة تُضر بالأفراد والمجتمعات.

**أخلاقيات الذكاء الاصطناعي (AI Ethics):**

تشير أخلاقيات الذكاء الاصطناعي إلى المبادئ، والقيم، والمعايير الأخلاقية التي تُنظّم تطوير أنظمة الذكاء الاصطناعي وانتشارها واستخدامها.

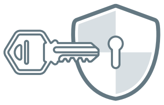
### أمثلة من العَالَم الواقعي على المخاوف الأخلاقية في مجال الذكاء الاصطناعي

### Real-World Examples of Ethical Concerns in AI

#### الخوارزميات التمييزية

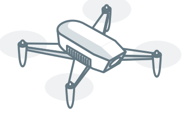
هناك مواقف تدل على أن أنظمة الذكاء الاصطناعي تميل إلى التحيُّز والتمييز ضد فئات معيّنة من البشر. على سبيل المثال، وجدت دراسة أجراها **المعهد الوطني للمعايير والتقنية** (National Institute of Standards and Technology) أن نِسب الخطأ في تقنية التعرُّف على الوجه تكون أعلى عند التعرُّف على وجوه الأشخاص ذوي البشرة الداكنة؛ مما قد يُؤدي إلى تحديد هويات خاطئة واعتقالات خاطئة. ومن الأمثلة الأخرى على ذلك استخدام خوارزميات الذكاء الاصطناعي في نظام العدالة الجنائية، إذ أظهرت الدراسات أن هذه الخوارزميات يُمكن أن تكون مُتحيِّزة ضد الأقليات مما يُؤدي إلى عقوبات أقسى.

#### انتهاك الخصُوصية



يُمكن أن تكون أنظمة الذكاء الاصطناعي التي تجمع البيانات وتُحلِّلها مصدر تهديد للخصُوصية الشخصية. على سبيل المثال: جمعت شركة استشارات سياسية في عام 2018 م بيانات الملايين من مستخدِمي **فيسبوك** (Facebook) دون موافقتهم واستخدمتها للتأثير على الحملات السياسية، وأثار هذا الحادث المخاوِف بشأن استخدام الذكاء الاصطناعي وتحليلات البيانات في التلاعب بالرأي العام، وانتهاك حقوق خصوصية الأفراد.

#### الأسلحة ذاتية التحكم



تطوير الأسلحة ذاتية التحكم التي يُمكن أن تعمل دون تدخل بشري له مخاوف أخلاقية بشأن استخدام الذكاء الاصطناعي في الحروب، حيث يرى فريق من المنتقدين أن هذه الأسلحة يُمكن أن تتخذ قرارات مصيرية دون إشراف بشري ويُمكن برمجتها لاستهداف مجموعات معيّنة من الناس، مما قد ينتهك القانون الإنساني الدولي، ويُؤدي إلى وقوع إصابات في صفوف المدنيين.

#### التسريح من الوظائف

أثار الاستخدام المتزايد للذكاء الاصطناعي **والأتمتة** (Automation) في مختلف الصناعات المخاوِف بشأن تسريح البشر من وظائفهم وتأثيره على سُبل عيش العاملين، فعلى الرغم من أن الذكاء الاصطناعي يُمكنه أن يُؤدي إلى تحسين الكفاءة والإنتاجية، إلا أنه يُمكن أن يُؤدي أيضًا إلى فقدان البشر لوظائفهم وتزايد عدم المساواة في الدخل؛ مما قد يكون له عواقب اجتماعية واقتصادية سلبية.

## التحيُّز والإنصاف في الذكاء الاصطناعي Bias and Fairness in AI

**تحيُّز الذكاء الاصطناعي (AI Bias):**

في مجال الذكاء الاصطناعي، يدل التحيُّز على ميل خوارزميات التعلُّم الآلي إلى إنتاج نتائج تحابي بدائل، أو فئات معيّنة، أو تظلمها بأسلوب منهجي؛ مما يؤدي إلى القيام بتنبؤات خاطئة وإلى احتمالية التمييز ضد مُنتَجات معيّنة أو فئات بشرية محدَّدة.

يُمكن أن يظهر **التحيُّز** (Bias) في أنظمة الذكاء الاصطناعي عندما تكون البيانات المستخدَمة لتدريب الخوارزميّة ناقصة التمثيل أو تحتوي على تحيُّزات أساسية، ويُمكن أن يظهر في أية بيانات تُمثِّلها مُخرَجات النظام، فعلى سبيل المثال لا الحصر: المُنتَجات والآراء والمجتمعات والاتجاهات كلها يمكن أن يظهر فيها التحيُّز.

يُعدُّ نظام التوظيف الآلي الذي يستخدِم الذكاء الاصطناعي لفحص المرشحين للوظائف من أبرز الأمثلة على الخوارزميّة المُتحيِّزة. افترض أن الخوارزميّة مُدرَّبة على بيانات مُتحيِّزة، مثل أنماط التوظيف التاريخية التي تُفضِّل مجموعات ديموغرافية معيّنة، ففي هذه الحالة قد يعمل الذكاء الاصطناعي على استمرار تلك التحيُّزات ويستبعد المرشّحين المؤهّلين بشكل غير عادل من بين المجموعات متجاهلًا الفئات غير الممثَّلة جيدًا في مجموعة البيانات. على سبيل المثال، افترض أن الخوارزميّة تُفضل المرشحين الذين التحقوا بجامعات النخبة، أو عملوا في شركات مرموقة، ففي هذه الحالة قد يلحق ذلك الضرر بالمرشحين الذين لم يحظوا بتلك الفُرص، أو الذين ينتمون إلى بيئات أقل حظًّا، ويُمكن أن يُؤدي ذلك إلى نقص التنوع في مكان العمل وإلى استمرارية عدم المساواة، ولذلك من المهم تطوير واستخدام خوارزميات توظيف للذكاء الاصطناعي تَستنِد على معايير عادلة وشفافة، وغير مُتحيِّزة.

يشير **الإنصاف** (Fairness) في الذكاء الاصطناعي إلى كيفية تقديم أنظمة الذكاء الاصطناعي لنتائج غير مُتحيِّزة وعلى معاملتها لجميع الأفراد والمجموعات مُعامَلة مُنصِفة، ولتحقيق الإنصاف في الذكاء الاصطناعي يتطلب ذلك تحديد التحيُّزات في البيانات والخوارزميات وعمليات اتخاذ القرار ومعالجتها. على سبيل المثال، تتمثّل إحدى طرائق تحقيق الإنصاف في الذكاء الاصطناعي في استخدام عملية تُسمى **إلغاء الانحياز** (Debiasing)، حيث يتم تحديد البيانات المُتحيِّزة وإزالتها أو تعديلها بما يضمن وصول الخوارزميّة إلى نتائج أكثر دقة دون تحيُّز.

**جدول 6.1: العوامل التي تُحدِّد تحيُّز أنظمة الذكاء الاصطناعي**

| | |
|---|---|
| بيانات التدريب المُتحيِّزة | تتعلّم خوارزميات الذكاء الاصطناعي من البيانات التي تُدرَّب عليها؛ فإذا كانت البيانات مُتحيِّزة أو ناقصة التمثيل، فقد تصل الخوارزميّة إلى نتائج مُتحيِّزة. على سبيل المثال، إذا تم تدريب خوارزميّة التعرُّف على الصور على مجموعة بيانات تحتوي في الغالب على أفراد ذوي بشرة فاتحة، فربما تُواجه صعوبة في التعرُّف بدقة على الأفراد ذوي البشرة الداكنة. |
| الافتقار إلى التنوُّع في فِرق التطوير | إذا لم يكن فريق التطوير متنوعًا ولا يُمثِّل نطاقًا واسعًا من الفئات الثقافية والتقنية، فقد لا يتعرَّف على التحيُّزات الموجودة في البيانات أو الخوارزميّة، ويؤدي الفريق الذي يتكون من أفراد من منطقة جغرافية أو ثقافة معيّنة إلى عدم مراعاة المناطق أو الثقافات الأخرى التي قد تكون مُمثَّلة في البيانات المُستخدَمة لتدريب نموذج الذكاء الاصطناعي. |
| الافتقار إلى الرقابة والمسؤولية | يُمكن أن يُؤدي الافتقار إلى الرقابة والمسؤولية في تطوير أنظمة الذكاء الاصطناعي ونشرها إلى ظهور التحيُّز، فإذا لم تُطبِّق الشركات والحكومات آليات رقابة ومُساءلة مناسبة، فإنّ ذلك قد يؤدي إلى عدم تنفيذ اختبار التحيُّز في أنظمة الذكاء الاصطناعي وربما لا يكون هناك مجال لإنصاف الأفراد أو المجتمعات المتضررة من النتائج المُتحيِّزة. |
| الافتقار إلى الخبرة والمعرفة لدى فريق التطوير | قد لا تُحدِّد فِرق التطوير التي تفتقر إلى الخبرة مؤشرات التحيُّز في بيانات التدريب أو تُعالجها، كما أن الافتقار إلى المعرفة في تصميم نماذج الذكاء الاصطناعي واختبارها لتحقيق العدالة ربما يُؤدي إلى استمرارية التحيُّزات القائمة. |



وزارة التعليم
Ministry of Education
2025 - 1447

# الحدّ من التحيُّز وتعزيز الإنصاف في أنظمة الذكاء الاصطناعي
# Reducing Bias and Promoting Fairness in AI Systems

### البيانات المتنوعة والمُمثّلة
يُقصد بذلك استخدام البيانات التي تعكس تنوع المجموعة التي يتم تمثيلها، كما أنه من المهم مراجعة وتحديث البيانات المُستخدَمة لتدريب أنظمة الذكاء الاصطناعي بانتظام؛ للتأكد من أنها ما زالت ملائمة وغير مُتحيِّزة.

### تقنيات إلغاء الانحياز
تتضمن أساليب إلغاء الانحياز تحديد وإزالة البيانات المُتحيِّزة من أنظمة الذكاء الاصطناعي؛ لتحسين معايير الدقة والإنصاف، فتشمل هذه التقنيات مثلًا: **زيادة العينات** (Oversampling) أو **تقليل العينات** (Undersampling) أو **زيادة البيانات** (Data Augmentation) لضمان تعرُّض نظام الذكاء الاصطناعي لنقاط بيانات مختلفة.

### القابلية للتفسير والشفافية
إنّ جعل أنظمة الذكاء الاصطناعي أكثر شفافية وأكثر قابلية للتفسير يمكنه أن يساعد في تقليص مستوى التحيُّز من خلال السماح للمُستخدِمين بفهم كيفية اتخاذ النظام للقرارات، ويتضمن ذلك توضيح عملية اتخاذ القرار والسماح للمُستخدِمين باستكشاف مُخرَجات النظام واختبارها.

### التصميم المعتمد على إشراك الإنسان
يُمكن أن يساهم إشراك العنصر البشري في حلقة تصميم أنظمة الذكاء الاصطناعي في التقليل من التحيُّز، وذلك بالسماح للبشر بالتدخل وتصحيح مُخرَجات النظام عند الضرورة، ويشمل ذلك تصميم أنظمة ذكاء اصطناعي بها مرحلة للتغذية الراجعة تُمكِّن البشر من مراجعة قرارات النظام والموافقة عليها.

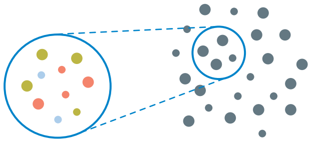
### المبادئ الأخلاقية
تعني دمج المبادئ الأخلاقية مثل: الإنصاف والشفافية والمساءلة، في تصميم وتنفيذ أنظمة الذكاء الاصطناعي، من أجل ضمان تطوير تلك الأنظمة واستخدامها بشكل أخلاقي ومسؤول، وذلك بوضع إرشادات أخلاقية واضحة لاستخدام أنظمة الذكاء الاصطناعي ومراجعة هذه الإرشادات بانتظام وتحديثها عند الضرورة.

### المراقبة والتقييم بانتظام
تُعدُّ المراقبة والتقييم بشكل دوري لأنظمة الذكاء الاصطناعي أمرًا ضروريًا لتحديد التحيُّز وتصحيحه، ويتضمن ذلك اختبار مُخرَجات النظام وإجراء عمليات تدقيق منتظمة؛ للتأكد من أن النظام يعمل بشكل عادل ودقيق.

### تقييم تغذية المُستخدِم الراجعة
يُمكن أن تساعد التغذية الراجعة التي يقدمها المُستخدِم في تحديد التحيُّز في النظام؛ لأن المُستخدِمين غالبًا ما يكونون أكثر وعيًا بتجاربهم، ويُمكنهم تقديم رؤى عن التحيُّز المحتمل أفضل مما يُمكن أن تقدمه خوارزميات الذكاء الاصطناعي. على سبيل المثال، يُمكن أن يقدِّم المُستخدِمون تغذية راجعة عن رؤيتهم لأداء نظام الذكاء الاصطناعي أو تقديم اقتراحات مفيدة لتحسين النظام وجعله أقل تحيُّزًا.

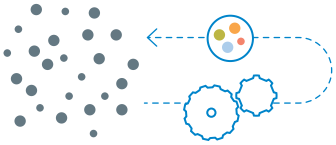
**زيادة العينات (Oversampling):**

تُشير زيادة العينة في تعلُّم الآلة إلى زيادة عدد عيّنات فئة ما داخل مجموعة بيانات لتحسين دقة النموذج، ويكون ذلك بواسطة المضاعفة العشوائية للعيّنات الموجودة في الفئة أو توليد عيّنات جديدة من الفئة نفسها.



**تقليل العينات (Undersampling):**

تقليل العينة هو عملية تقليل حجم مجموعة البيانات بحذف مجموعة فرعية من بيانات الفئة الأكبر للتركيز على العيّنات الأكثر أهمية. ويكون ذلك مفيدًا بشكل خاص إذا كانت مجموعة البيانات تفتقر إلى التوازن بين الفئات أو بين مجموعاتها المختلفة.



**زيادة البيانات (Data Augmentation):**

زيادة البيانات هي عملية توليد بيانات تدريب جديدة من البيانات الموجودة لتعزيز أداء نماذج تعلُّم الآلة، ومن الأمثلة على ذلك: **قلب الصور** (Image Flipping) وتدويرها وقصها وتغيير ألوانها وتحويلها **تحويلًا تآلفيًا** (Affine Transformation) والتشويش عليها.


وزارة التعليم
Ministry of Education
2025 - 1447

## مشكلة المسؤولية الأخلاقية في الذكاء الاصطناعي
## The Problem of Moral Responsibility in AI

تُعدُّ مشكلة المسؤولية الأخلاقية عند استخدام أنظمة الذكاء الاصطناعي المتقدمة قضية مُعقَّدة ومتعددة الجوانب، وقد حظيت باهتمام كبير في السنوات الأخيرة.

تتمثّل إحدى التحديات الرئيسة لأنظمة الذكاء الاصطناعي المتقدمة في قدرتها على اتخاذ القرارات والقيام بإجراءات يُمكن أن يكون لها عواقب إيجابية أو سلبية كبيرة على الأفراد والمجتمع، ورغم ذلك، لا يكون الطرف الذي يجب تحميله المسؤولية الأخلاقية عن هذه النتائج محدّدًا دائمًا.

هناك رأي يقول: إن مطوِّري ومصمِّمي أنظمة الذكاء الاصطناعي يجب أن يتحملوا المسؤولية عن أي نتائج سلبية تَنتُج عن استخدامها، ويُؤكِّد هذا الرأي على أهمية ضمان تصميم أنظمة ذكاء اصطناعي تُراعي الاعتبارات الأخلاقية وتُحمِّل المطوِّرين المسؤولية عن أي ضرر قد تسببه اختراعاتهم.

ويرى آخرون أن المسؤولية عن نتائج الذكاء الاصطناعي هي مسؤولية مشتركة بين أصحاب المصلحة بما فيهم صُنّاع السياسات، والمنظمين ومُستخدِمي التقنية، ويسلط هذا الرأي الضوء على أهمية ضمان استخدام أنظمة الذكاء الاصطناعي بطرائق تتماشى مع المبادئ الأخلاقية، وتقييم المخاطر المرتبطة باستخدامها وإدارتها بعناية.

وهناك رأي ثالث يقول: إن أنظمة الذكاء الاصطناعي هي"ذاتٌ مسؤولة" لديها حسٌّ أخلاقي ومسؤولة عن أفعالها، وتقول هذه النظرية: إنّ أنظمة الذكاء الاصطناعي المُتقدِّمة يُمكن أن تتمتع بالفاعلية والاستقلالية؛ مما يجعلها أكثر من مجرد أدوات، كما تتطلب منها أن تكون مسؤولة عن أفعالها، إلا أن لهذه النظرية عدة مشكلات.

تستطيع أنظمة الذكاء الاصطناعي أن تُصدِر أحكامًا وأن تتصرف من تلقاء نفسها، ولكنها ليست "ذاتًا مسؤولة" لديها حسٌّ أخلاقي وذلك للأسباب التالية:

أولًا: أن أنظمة الذكاء الاصطناعي تفتقر إلى الوعي والخبرات الذاتية؛ مما يُعدُّ سمة أساسية من سمات "الذات المسؤولة" التي لديها حسٌّ أخلاقي، وفي العادة تتضمن الفاعلية الأخلاقية القدرة على التفكير في المُثُل العليا للفرد وأفعاله.

ثانيًا: يقوم الأشخاص بتدريب أنظمة الذكاء الاصطناعي على اتباع قواعد وأهداف محدَّدة؛ مما يحدُّ من حكمها الأخلاقي، ويُمكن لأنظمة الذكاء الاصطناعي تكرار اتخاذ القرارات الأخلاقية، مع افتقارها للإرادة الحُرة والاستقلالية الشخصية.

وأخيرًا، فإن مُنشِئي أنظمة الذكاء الاصطناعي والقائمين على نشرها هم المسؤولون عن أفعالهم، ويُمكن لأنظمة الذكاء الاصطناعي أن تُساعد في اتخاذ القرارات الأخلاقية، على الرغم من أنها ليست "ذاتًا مسؤولة" لديها حسٌّ أخلاقي.

وزارة التعليم
Ministry of Education
2025 - 1447

# الشفافية وقابلية التفسير في الذكاء الاصطناعي ومشكلة الصندوق الأسود

## Transparency and Explainability in AI and the Black-Box Problem

تكمن مشكلة الصندوق الأسود في الذكاء الاصطناعي في التحدي المُتمثِّل في فهم كيفية عمل **نظام قائم على الذكاء الاصطناعي** (AI-Based System) باتخاذ القرارات أو إنتاج المُخرَجات؛ مما قد يُصعِّب الوثوق بالنظام أو تفسيره أو تحسينه، وربما يؤثر الافتقار إلى الانفتاح وإلى قابلية التفسير على ثقة الناس في النموذج. تتزايد هذه التحديات بوجه خاص في مجال التشخيص الطبي، والأحكام التي تصدرها المركبات ذاتية القيادة. تُعدُّ التحيُّزات في نماذج تعلُّم الآلة إحدى المخاوف الأخرى المتعلقة بنماذج الصندوق الأسود، كما أن التحيُّزات الموجودة في البيانات التي يتم تدريب هذه النماذج عليها يُمكن أن تُؤدي إلى نتائج غير عادلة أو عنصرية. بالإضافة إلى ذلك، ربما يكون من الصعب تحديد المسؤولية عن القرارات التي يتخذها نموذج الصندوق الأسود؛ حيث يصعب تحميل أي شخص المسؤولية عن تلك القرارات لا سيما مع وجود الحاجة إلى الرقابة البشرية، كما هو الحال في أنظمة الأسلحة ذاتية التحكم. إن الافتقار إلى الشفافية في عملية اتخاذ القرار باستخدام الذكاء الاصطناعي يُصعِّب تحديد مشكلات النموذج وحلّها، كما أن عدم معرفة الطريقة التي يتخذ بها النموذج قراراته تجعل من الصعب إجراء التحسينات والتأكد من أنها تعمل بطريقة صحيحة، وهناك استراتيجيات عديدة لمعالجة مشكلة الصندوق الأسود في الذكاء الاصطناعي. تتمثّل إحدى تلك الاستراتيجيات في استخدام تقنيات ذكاء اصطناعي قابلة للتفسير لجعل نماذج تعلُّم الآلة أكثر شفافية وأكثر قابلية للتفسير، وقد تشمل هذه التقنيات: **مُفسرات اللغات الطبيعية** (Natural Language Explanation) أو تصوير البيانات للمساعدة في فهم عملية اتخاذ القرار، وهناك أسلوب آخر يتمثل في استخدام نماذج تعلُّم الآلة الأكثر قابلية للتفسير مثل: **أشجار القرار** (Decision Trees) أو **الانحدار الخطي** (Linear Regression)، وربما تكون هذه النماذج أقل تعقيدًا وأسهل في الفهم، ولكنها قد لا تكون قوية أو دقيقة مثل النماذج الأكثر تعقيدًا. تُعدُّ معالجة مشكلة الصندوق الأسود في الذكاء الاصطناعي أمرًا مهمًّا لبناء الثقة في نماذج تعلُّم الآلة وضمان استخدامها بأسلوب أخلاقي وعادل.

**نظام الصندوق الأسود (Black-Box System):**

هو نظام لا يكشف عن طرائق عمله الداخلية للبشر؛ إذ تتم التغذية بالمُدخَلات، ليتم إنتاج المُخرَجات دون معرفة طريقة عملها، كما هو موضَّح في الشكل 6.1.

شكل 6.1: نظام الصندوق الأسود

## طرائق تعزيز شفافية نماذج الذكاء الاصطناعي وقابليتها للتفسير

## Methods for Enhancing the Transparency and Explainability of AI Models

### النموذج المحايد المحلي القابل للتفسير والشرح

**النموذج المحايد المحلي القابل للتفسير والشرح** (Local Interpretable Model-Agnostic Explanations – LIME) تم استخدامه مسبقًا في مهام **معالجة اللغات الطبيعية** (NLP)، وتقوم هذه التقنية بتوليد تفسيرات محلية لتنبؤات مفردة يتم إجراؤها بواسطة نموذج، وتُنشئ هذه التفسيرات نموذجًا أبسط وقابلًا للتفسير يقارب نموذج الصندوق الأسود المُعقَّد حول تنبؤ محدَّد، ثم يُستخدم هذا النموذج البسيط لشرح كيف توصّل إلى قراره بشأن هذا التنبؤ المحدَّد. تتمثّل ميزة هذه التقنية في أنها تُوفر تفسيرات يُمكن للإنسان قراءتها، وبالتالي يُمكن لأصحاب المصلحة غير المتخصصين فهمها بسهولة؛ حتى فيما يتعلق بالنماذج المُعقَّدة مثل: **الشبكات العصبية العميقة** (Deep Neural Networks).

### تفسيرات شابلي الإضافية

**تفسيرات شابلي الإضافية** (SHapley Additive exPlanations – SHAP) هي طريقة أخرى لتفسير مُخرَجات نماذج تعلُّم الآلة، وتعتمد على المفهوم الخاص بقيم شابلي من **نظرية الألعاب** (Game Theory) وتُخصِّص قيمة (أو وَزنًا) لكل خاصية

مساهمة في التنبؤ. يُمكن استخدام الطريقة مع أي نموذج، كما تقدم تفسيراتٍ في شكل درجات تبيّن أهميّة الخصائص، مما يُمكن أن يساعد في تحديد الخصائص الأكثر تأثيرًا في مُخرَجات النموذج.

وهناك تقنية أخرى لتحسين قابلية تفسير الذكاء الاصطناعي مثل: أشجار القرار وقواعد القرار، وهي نماذج قابلة للتفسير يُمكن تصويرها بسهولة، حيث تقوم أشجار القرار بتقسيم **فضاء الخصائص** (Feature Space) بناءً على الخاصية الأكثر دلالة، وتقدِّم قواعد واضحة لاتخاذ القرارات، وتُعدُّ أشجار القرار مفيدة بشكل خاص عندما تتخذ البيانات شكل الجداول ويكون هناك عدد محدود من الخصائص. ولكن هذه النماذج محدودة أيضًا؛ لأن قابلية تفسير شجرة القرار التي تم إنشاؤها تتناسب تناسبًا عكسيًّا مع حجم الشجرة. على سبيل المثال، من الصعب فهم الأشجار التي تتكون من آلاف العقد ومئات المستويات. وأخيرًا، هناك أسلوب آخر يستخدِم تقنيات مثل: **وكلاء الذكاء الاصطناعي** (Artificial Intelligence Agents) أو **تحليل الحساسية** (Sensitivity Analysis) للمساعدة في فهم كيفية تأثير تغيير المُدخَلات أو الافتراضات على مُخرَجات النموذج، ويُمكن أن يكون هذا الأسلوب مفيدًا بشكل خاص في تحديد مصادر الغموض في النموذج وفي فهم حدوده.

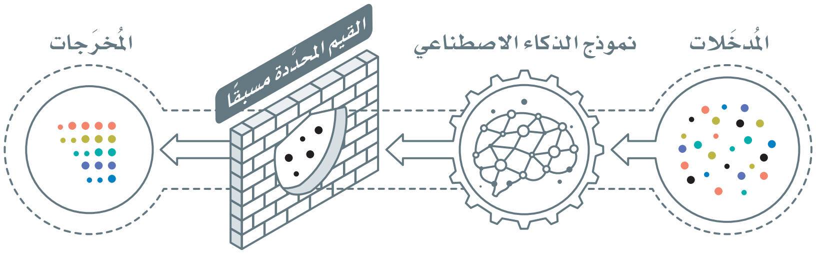
## الاستدلال القائم على القِيم في أنظمة الذكاء الاصطناعي

## Value-Based Reasoning in AI Systems

يتمثّل الهدف من ذلك في إنشاء أنظمة ذكاء اصطناعي أكثر اتساقًا مع القيم والأخلاقيات البشرية؛ بحيث تتعامل هذه الأنظمة بطرائق مفيدة ومنصفة ومسؤولة. تتضمن الخطوة الأولى في الاستدلال القائم على القِيم، فهمَ وتمثيل القِيم الأخلاقية داخل أنظمة الذكاء الاصطناعي، حيث يجب أن تكون هذه الأنظمة قادرة على تفسير وتوطين القيم أو المبادئ التوجيهية الأخلاقية التي يُقدمها منشؤها البشريون أو أصحاب المصلحة، وقد تتضمن هذه العملية التعلُّم من الأمثلة أو التغذية الراجعة البشرية أو القواعد الواضحة، وعندما تفهم أنظمة الذكاء الاصطناعي هذه القيم بوضوح، يُمكنها أن تقوم بمواءمة أفعالها بطريقة أفضل مع المبادئ الأخلاقية المنشودة.

**الاستدلال القائم على القِيم (Value-Based Reasoning):**

الاستدلال القائم على القِيم في أنظمة الذكاء الاصطناعي يشير إلى العملية التي يستخدِمها وكلاء الذكاء الاصطناعي لاتخاذ قرارات أو استخلاص نتائج بناءً على مجموعة محدَّدة مسبقًا من القيم أو المبادئ أو الاعتبارات الأخلاقية.




شكل 6.2: تمثيل للاستدلال القائم على القِيم

يُركز الجانب الثاني من جوانب الاستدلال القائم على القِيم على تقييم القرارات أو الأفعال بناءً على **القِيم التي وُطنّت** (Internalized Values)، ويجب أن تقوم أنظمة الذكاء الاصطناعي بتقييم النتائج المحتملة للقرارات أو الإجراءات المختلفة بالنظر في عواقب كل خيار ومخاطره وفوائده، كما يجب أن تأخذ عملية التقييم هذه في الاعتبار القيم الأساسية التي تم تصميم نظام الذكاء الاصطناعي لدعمها، مما يضمن أن يتخذ النظام خيارات مستنيرة ومتوافقة مع القيم.

وأخيرًا، يتطلب الاستدلال القائم على القِيم من أنظمة الذكاء الاصطناعي اتخاذ قرارات تتماشى مع القِيم الراسخة، فبعد تقييم الخيارات المختلفة ونتائجها المحتملة، يجب على نظام الذكاء الاصطناعي أن ينتقي القرار أو الإجراء الذي يُمثِّل المبادئ والأهداف الأخلاقية التي صُمِّم لاتباعها، فمن خلال اتخاذ قرارات متوافقة مع القِيم، يمكن **لوكلاء الذكاء الاصطناعي** (AI Agents) التصرف بطرائق تتفق مع المبادئ التوجيهية الأخلاقية التي وضعها مُنشؤها؛ مما يعزّز السلوك المسؤول والمفيد. على سبيل المثال: تُستخدم أنظمة الذكاء الاصطناعي في الرعاية الصحية للمساعدة في اتخاذ

وزارة التعليم
Ministry of Education
2025 - 1447

قرارات التشخيص والعلاج، حيث يجب أن تكون هذه الأنظمة قادرة على التفكير في الآثار الأخلاقية المترتبة على العلاجات المختلفة مثل: الآثار الجانبية المحتملة أو التأثير على جودة الحياة، ومن ثَمّ تتخذ قرارات تُعطي الأولوية لسلامة المريض، ومن الأمثلة الأخرى: أنظمة الذكاء الاصطناعي المُستخدَمة في التمويل للمساعدة في اتخاذ قرارات الاستثمار، حيث يجب أن تكون هذه الأنظمة قادرة على أن تُفكر في الآثار الأخلاقية المترتبة على الاستثمارات المختلفة، كالتأثير على البيئة أو على الرعاية الاجتماعية، وبالتالي تتخذ القرارات التي تتماشى مع قِيم المستثمر.

يجب أن ندرك أن المسؤولية لا تقع بأكملها على عاتق نظام الذكاء الاصطناعي، بل إنها مسؤولية مشتركة بين الذكاء الاصطناعي والخبراء البشريين، فنظام الذكاء الاصطناعي يساعد في اتخاذ القرار بأن يُلخِّص الحالة ويقدِّم الخيارات أو العروض للمُستخدِم الخبير الذي يتخذ القرار النهائي؛ مما يؤكد أن الخبير البشري هو المتحكم والمسؤول عن النتيجة النهائية، في ظل الاستفادة من الأفكار والتحليلات التي يُوفرها نظام الذكاء الاصطناعي.

## الذكاء الاصطناعي وتأثيره على البيئة AI and Environmental Impact

إن تأثير الذكاء الاصطناعي على البيئة وعلى علاقتنا بها مُعقَّد ومتعدد الأوجه.

شكل 6.3: تحليل الذكاء الاصطناعي لكميات ضخمة من البيانات

**فوائده المحتملة**

يُمكن للذكاء الاصطناعي أن يساعد في فهم التحديات البيئية والتعامل معها بشكل أفضل مثل: تغير المناخ، والتلوث، وفقدان التنوع البيولوجي، ويُمكنه أن يساعد في تحليل كميات هائلة من البيانات والتنبؤ بتأثير الأنشطة البشرية المختلفة على البيئة، ويُمكنه كذلك أن يساعد في تصميم أنظمة أكثر كفاءة واستدامة مثل: أنظمة شبكات الطاقة، والزراعة، والنقل، والمباني.

**أخطاره أو أضراره المُحتملة**

هناك مخاوف من تأثير الذكاء الاصطناعي نفسه على البيئة؛ إذ يتطلب تطوير أنظمة الذكاء الاصطناعي واستخدامها قدرًا كبيرًا من الطاقة والموارد؛ مما قد يُسهِّم في انبعاث غازات تُفاقِم من مشكلة الاحتباس الحراري وغيرها من الآثار البيئية. على سبيل المثال، قد يتطلب تدريب نموذج واحد للذكاء الاصطناعي قدرًا من الطاقة يعادل ما تستهلكه العديد من السيارات طوال حياتها. بالإضافة إلى ذلك، يمكن أن يساهم إنتاج المُكوِّنات الإلكترونية المُستخدَمة في تصنيع أنظمة الذكاء الاصطناعي في تلوث البيئة مثل: استخدام المواد الكيميائية السامة وتوليد النفايات الإلكترونية.

شكل 6.4: تتطلب أنظمة الذكاء الاصطناعي كميات هائلة من الطاقة والموارد

علاوة على ذلك، يُمكن أن يغير الذكاء الاصطناعي علاقتنا بالبيئة بطرائق ليست إيجابية دائمًا، فقد يُؤدي استخدام الذكاء الاصطناعي في الزراعة إلى ممارسات زراعية مكثَّفة ومركِّزة على الصناعة؛ مما يؤثر سلبًا على صحة التربة والتنوع البيولوجي. بالمثل، ربما يُؤدي استخدام الذكاء الاصطناعي في النقل إلى زيادة الاعتماد على السيارات وأساليب النقل الأخرى؛ مما يُسهِّم في تلوث الهواء وتدمير البيئات الطبيعية التي تسكنها الكائنات الحية.

**الخاتمة**

بوجه عام، يعتمد تأثير الذكاء الاصطناعي على البيئة وعلاقتنا بها على كيفية تطوير أنظمة الذكاء الاصطناعي واستخدامها، ومن المهم النظر في التأثيرات البيئية المحتملة للذكاء الاصطناعي وتطوير أنظمته واستخدامها بطرائق تُعطي الأولوية للاستدامة والكفاءة وسلامة كوكب الأرض.

## الأُطر التنظيمية ومعايير الصناعة
## Regulatory Frameworks and Industry Standards

تلعب الأُطر التنظيمية ومعايير الصناعة دورًا مهمًّا في تعزيز تطبيقات الذكاء الاصطناعي الأخلاقية، فبإمكان التنظيمات المُساعِدة أن تضمن تَحمُّل المنظمات التي تقوم بتطوير واستخدام أنظمة الذكاء الاصطناعي المسؤولية عن أفعالها عن طريق تحديد توقُّعات وعواقب واضحة لعدم الامتثال، وبإمكان التنظيمات والمعايير أن تُحفز المنظمات على إعطاء الأولوية للاعتبارات الأخلاقية عند تطوير واستخدام أنظمة الذكاء الاصطناعي.

**الشفافية**

يُمكن أن تعزِّز التنظيمات والمعايير الشفافية في أنظمة الذكاء الاصطناعي بمطالبة المؤسسات بالكشف عن كيفية عمل أنظمتها وعن البيانات التي تستخدِمها، ويُمكن أن يساعد ذلك في بِناء الثقة مع أصحاب المصلحة وتقليل المخاوف من التحيُّزات المحتملة أو التمييز المحتمل في أنظمة الذكاء الاصطناعي.

**تقييم المخاطر**

يُمكن تقليل مخاطر العواقب غير المقصودة أو النتائج السلبية الناتجة عن استخدام الذكاء الاصطناعي بوضع التنظيمات والمعايير المناسبة، وذلك بمطالبة المنظمات بإجراء تقييمات للمخاطر، وهذا يعني تحديد المخاطر والأخطار المحتملة وتنفيذ ضمانات مناسبة، مما يُمكِّن التنظيمات والمعايير من المساعدة في تقليل الأضرار المحتملة على الأفراد والمجتمع.

**تطوير ونشر أُطر عمل واضحة للذكاء الاصطناعي**

يُمكن أن تشجِّع التنظيمات والمعايير الابتكار بتوفير إطار عمل واضح لتطوير أنظمة الذكاء الاصطناعي واستخدامها؛ إذ أن استخدام التنظيمات والمعايير لتأسيس فرص متكافئة وتقديم التوجيه بخصوص الاعتبارات الأخلاقية يُمكن أن يساعد المنظمات على تطوير أنظمة الذكاء الاصطناعي ونشرها بطرائق تتفق مع القيم الأخلاقية والاجتماعية. تلعب الأُطر التنظيمية ومعايير الصناعة دورًا مهمًّا في تعزيز تطبيقات الذكاء الاصطناعي الأخلاقية، وذلك بتوفير إرشادات وحوافز واضحة للمؤسسات حتى تُعطي الأولوية للاعتبارات الأخلاقية والتنظيمات والمعايير؛ مما يضمن تطوير أنظمة الذكاء الاصطناعي واستخدامها بطرائق تتماشى مع القيم الاجتماعية والأخلاقية.

## التنمية المستدامة للذكاء الاصطناعي في المملكة العربية السعودية
## Sustainable AI Development in the Kingdom of Saudi Arabia

من المتوقَّع أن تصبح تقنيات الذكاء الاصطناعي وأنظمته أحد العوامل الرئيسة التي تُؤدي إلى إحداث خلل في القطاعات المالية في العديد من البلدان، وقد تؤثر بشكل كبير على سوق العمل، ومن المتوقَّع في السنوات القادمة أن يصبح حوالي 70 % من الأعمال الروتينية التي يقوم بها العمال مؤتمتة بالكامل. كما أنه من المتوقَّع أن تخلق صناعة الذكاء الاصطناعي سبعة وتسعين مليون وظيفة جديدة وتضيف ستة عشر تريليون دولار أمريكي إلى الناتج المحلي الإجمالي العَالَمي.

SDAIA
الهيئة السعودية للبيانات والذكاء الاصطناعي
Saudi Data & AI Authority

لقد طوَّرت **الهيئة السعودية للبيانات والذكاء الاصطناعي** (Saudi Data and Artificial Intelligence Authority - SDAIA) أهدافًا استراتيجية للمملكة لاستخدام تقنيات الذكاء الاصطناعي المُستدامة في تنمية المملكة، وستكون المملكة العربية السعودية مركزًا عالميًا للبيانات والذكاء الاصطناعي، كما أن المملكة استضافت أول قمة عالمية لهُ، حيث يُمكن للقادة والمبتكرين مناقشة مستقبل الذكاء الاصطناعي وتشكيله لصالح المجتمع. أما الهدف الآخر فيتمثل في تحويل القوى العاملة في المملكة من خلال تطوير البيانات المحلية ودعم المواهب في الذكاء الاصطناعي. وبما أن الذكاء الاصطناعي يقوم بتحويل أسواق العمل عالميًا، فإن معظم القطاعات تحتاج إلى تكييف البيانات والذكاء الاصطناعي ودمجها في التعليم والتدريب المهني والمعرِفة العامة، وبذلك يُمكن أن تكتسب المملكة العربية السعودية ميزة تنافسية من حيث التوظيف والإنتاجية والابتكار.

أما الهدف النهائي فيتمثّل في جذب الشركات والمستثمرين عن طريق أُطر عمل وحوافز تنظيمية مرنة ومستقرة، حيث ستركز الأنظمة على تطوير سياسات ومعايير للذكاء الاصطناعي، بما فيها استخدامه بشكل أخلاقي. وسيعمل إطار العمل على تعزيز التطوير الأخلاقي لأبحاث وحلول الذكاء الاصطناعي ودعمه في ظل توفير إرشادات ومعايير لحماية البيانات والخصوصية؛ مما سيُوفر الاستقرار والتوجيه لأصحاب المصلحة العاملين في المملكة.

**مثال**

NEOM

تُخطط المملكة العربية السعودية لاستخدام أنظمة وتقنيات الذكاء الاصطناعي كأساس لمشروعي المدينتين العملاقتين نيوم (**NEOM**) وذا لاين (**THE LINE**). مشروع نيوم هو مدينة مستقبلية سيتم تشغيلها بالطاقة النظيفة، وبها أنظمة نقل متطورة، وتقدِّم خدمات ذات تقنية عالية، وستكون منصة للتقنيات المتطورة، بما في ذلك الذكاء الاصطناعي، وستستخدِم حلول المُدن الذكية؛ لتحسين استهلاك الطاقة وإدارة حركة المرور والخدمات المتقدمة الأخرى. وسيتم استخدام أنظمة الذكاء الاصطناعي فيها؛ لتحسين جودة الحياة للسكان ولتعزيز الاستدامة.

وبالمثل، ستكون مدينة ذا لاين مدينة خطية خالية من الكربون مبنية بتقنيات الذكاء الاصطناعي، وستستخدِم أنظمة الذكاء الاصطناعي لأتمتة بنيتها التحتية وأنظمة النقل فيها؛ مما يجعل حياة المقيمين فيها تتسم بالسلاسة والكفاءة، وستكون الطاقة التي ستُشغِّل المدينة طاقة نظيفة، كما أن الأولوية ستكون للمعيشة المستدامة، وسيتم استخدام الأنظمة التي تعمل بالذكاء الاصطناعي؛ لمراقبة استخدام الطاقة وتحسينها وانسيابية حركة المرور والخدمات المتقدمة الأخرى.

وبوجه عام، ستلعب أنظمة الذكاء الاصطناعي وتقنياته دورًا حاسمًا في تطوير مشروعي هاتين المدينتين العملاقتين، وتمكينهما من أن تصبحا مدينتين مستدامتين من مُدن المستقبل تتسمان بالكفاءة والابتكار.

## الإرشادات العالمية لأخلاقيات الذكاء الاصطناعي International AI Ethics Guidelines

كما هو موضَّح في الجدول التالي، طوَّرت منظمة **اليونسكو** (UNESCO) وثيقة إرشادية توضِّح بالتفصيل القِيم والمبادئ التي يجب الالتزام بها عند تطوير أنظمة وتقنيات الذكاء الاصطناعي الجديدة.

**جدول 6.2: قِيم ومبادئ أخلاقيات الذكاء الاصطناعي**

| القِيم | المبادئ |
|---|---|
| • احترام كرامة الإنسان وحمايتها وتعزيزها، وحفظ حريته وحقوقه الأساسية.<br>• ازدهار البيئة والنظام البيئي.<br>• ضمان التنوع والشمولية.<br>• العيش في انسجام وسلام. | • التناسب وعدم الإضرار.<br>• السلامة والأمن.<br>• الإنصاف وعدم التمييز.<br>• الاستدامة.<br>• الخصوصية.<br>• الرقابة البشرية والعزيمة.<br>• الشفافية وقابلية التفسير.<br>• المسؤولية والمساءلة.<br>• الوعي والتثقيف.<br>• الحوكمة والتعاون القائمان على تعدُّد أصحاب المصلحة. |


وزارة التعليم
Ministry of Education
2025 - 1447

# تمرينات

1

| حَدِّد الجملة الصحيحة والجملة الخاطئة فيما يلي: | صحيحة | خاطئة |
|---|---|---|
| 1. تهتم أخلاقيات الذكاء الاصطناعي بتطوير أنظمة الذكاء الاصطناعي فقط. | ⚪ | ⚪ |
| 2. من المحتمل أن يُؤدي الذكاء الاصطناعي والأتمتة إلى تسريح البشر من الوظائف. | ⚪ | ⚪ |
| 3. يُمكن أن يُؤدي الافتقار إلى التنوع في فِرق تطوير الذكاء الاصطناعي إلى عدم رؤية التحيُّزات أو عدم معالجتها. | ⚪ | ⚪ |
| 4. يُمكن أن يساعد دمج المبادئ الأخلاقية في أنظمة الذكاء الاصطناعي في ضمان تطويرها واستخدامها بطريقة مسؤولة. | ⚪ | ⚪ |
| 5. يتطلب التصميم المعتمد على إشراك الإنسان أن تعمل أنظمة الذكاء الاصطناعي دون أي تدخل بشري. | ⚪ | ⚪ |
| 6. تدل مشكلة الصندوق الأسود في الذكاء الاصطناعي على صعوبة فهم كيفية وصول خوارزميات الذكاء الاصطناعي إلى قراراتها أو تنبؤاتها. | ⚪ | ⚪ |
| 7. يُمكن تصميم نماذج الذكاء الاصطناعي لتكييف قراراتها أو نتائجها وفقًا للقِيم الأخلاقية الراسخة. | ⚪ | ⚪ |
| 8. استخدام الذكاء الاصطناعي على نطاق واسع له آثار إيجابية فقط على البيئة. | ⚪ | ⚪ |

2 صِف كيف يؤدي الذكاء الاصطناعي والأتمتة إلى تسريح البشر من وظائفهم.

________________________________________

________________________________________

________________________________________

________________________________________

________________________________________

________________________________________

________________________________________

3 اشرح كيف يمكن أن تساهم بيانات التدريب المُتحيِّزة في تحقيق نتائج ذكاء اصطناعي مُتحيِّزة.

4 عرِّف مشكلة الصندوق الأسود في أنظمة الذكاء الاصطناعي.

5 قارن بين الآثار الإيجابية والسلبية لأنظمة الذكاء الاصطناعي على البيئة.



وزارة التعليم
Ministry of Education
2025 - 1447

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الثاني

# التطبيقات الروبوتية 1

## إحداث ثورة في العَالَم باستخدام الروبوتية

## Revolutionizing the World with Robotics

الروبوتية هي مجال سريع النمو أحدثَ ثورة في طريقة عمل الناس وفي عيشهم وتفاعلهم مع بيئتهم وتطبيقاتها، وتشمل مجموعة واسعة من المجالات: بداية من التصنيع وحتى استكشاف الفضاء، ومن الإجراءات الطبية إلى تنظيف المنزل، ومن الترفيه إلى المهام العسكرية. وتتمثّل الميزة الرئيسة للروبوتية في قدرتها على أداء المهام المتكررة بدرجة عالية من الدقة والإتقان، حيث يُمكن أن تعمل الروبوتات بلا تعب ودون أخطاء؛ مما يجعلها مثالية للقيام بالمهام الخطرة أو التي يصعب على البشر القيام بها. على سبيل المثال، في العمليات المصنعية تُستخدم الروبوتات لأداء بعض المهام مثل: اللحام والطلاء وتجميع المُنتَجات، وفي المجال الطبي تُستخدم الروبوتات لإجراء العمليات الجراحية بدقة أكبر، وفي استكشاف الفضاء تُستخدم الروبوتات لاستكشاف ودراسة الكواكب البعيدة.

**الروبوتية (Robotics):**

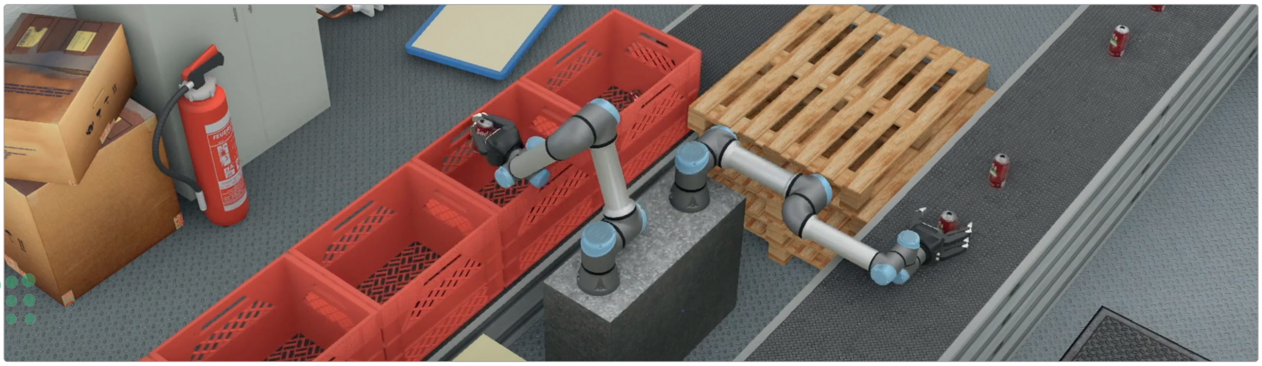
تهتم الروبوتية بدراسة الروبوتات، وهي آلات يمكنها أداء مجموعة متنوعة من المهام بطريقة مستقلة أو شبه مستقلة أو تحت تصرُّف البشر.

### الروبوتية والمُحاكِيات Robotics and Simulators

هناك تحديان مهمان في مجال الروبوتية هما: التكلفة والوقت اللازمان لبناء أجهزة الروبوت الماديّة واختبارها، وهنا يأتي دور **المُحاكِيات** (Simulators) التي تُستخدم على نطاق واسع في أبحاث الروبوتية وتعليمها وصناعتها؛ لأنها توفر طريقة فعّالة من حيث التكلفة، كما أنها آمنة لاختبار الروبوتات وتجربتها، حيث تتيح المُحاكِيات للمطوِّرين إنشاء بيئات افتراضية تُحاكي سيناريوهات العَالَم الحقيقي؛ مما يسمح لهم باختبار قدرات الروبوتات وأدائها في مجموعة متنوعة من المواقف، ويُمكنها محاكاة مختلف الظروف الجوية والتضاريس والعقبات التي قد تواجهها الروبوتات في العَالَم الحقيقي. كما يُمكن للمُحاكِيات أن تُحاكي التفاعلات بين الروبوتات المتعددة وبين الروبوتات والبشر؛ مما يسمح للمطوِّرين بدراسة وتحسين الطرائق التي تتفاعل بها الروبوتات مع بيئتها.

**المُحاكي (Simulator):**

برنامج يسمح للمطوِّرين باختبار تصميماتهم وخوارزمياتهم وتحسينها في عالم افتراضي قبل بِناء الروبوتات الماديّة.



شكل 6.5: محاكاة للأذرع الصناعية



وزارة التعليم
Ministry of Education
2025 - 1447

وهناك ميزة أخرى للمُحاكِيات تتمثّل في أنها تسمح للمطوِّرين بتعديل تصاميم وخوارزميات الروبوتات المختلفة، واختبارها بسهولة دون الحاجة إلى مُكوّنات ماديّة حاسوبية باهظة الثمن؛ حيث تسمح بالتكرار والتجريب بطريقة أسرع، مما يُؤدي إلى دورات تطوير أكثر سرعة وتصميمات أكثر كفاءة.

وبوجه عام، تُعدُّ الروبوتية مجالًا سريع النمو يتضمن مجموعة واسعة من التطبيقات والمُحاكِيات التي تلعب دورًا مهمًّا في تطوير الروبوتات عن طريق السماح للمطوِّرين باختبار تصاميم الروبوتات وخوارزمياتها، وتحسينها بطريقة آمنة وغير مُكلفة، ومع استمرار تقدُّم التقنية، فمن المتوقَّع أن تنمو تطبيقات الروبوتية واستخدام المُحاكِيات، مما يمهّد الطريق لعَالَم أكثر أتمتةً وترابطًا.

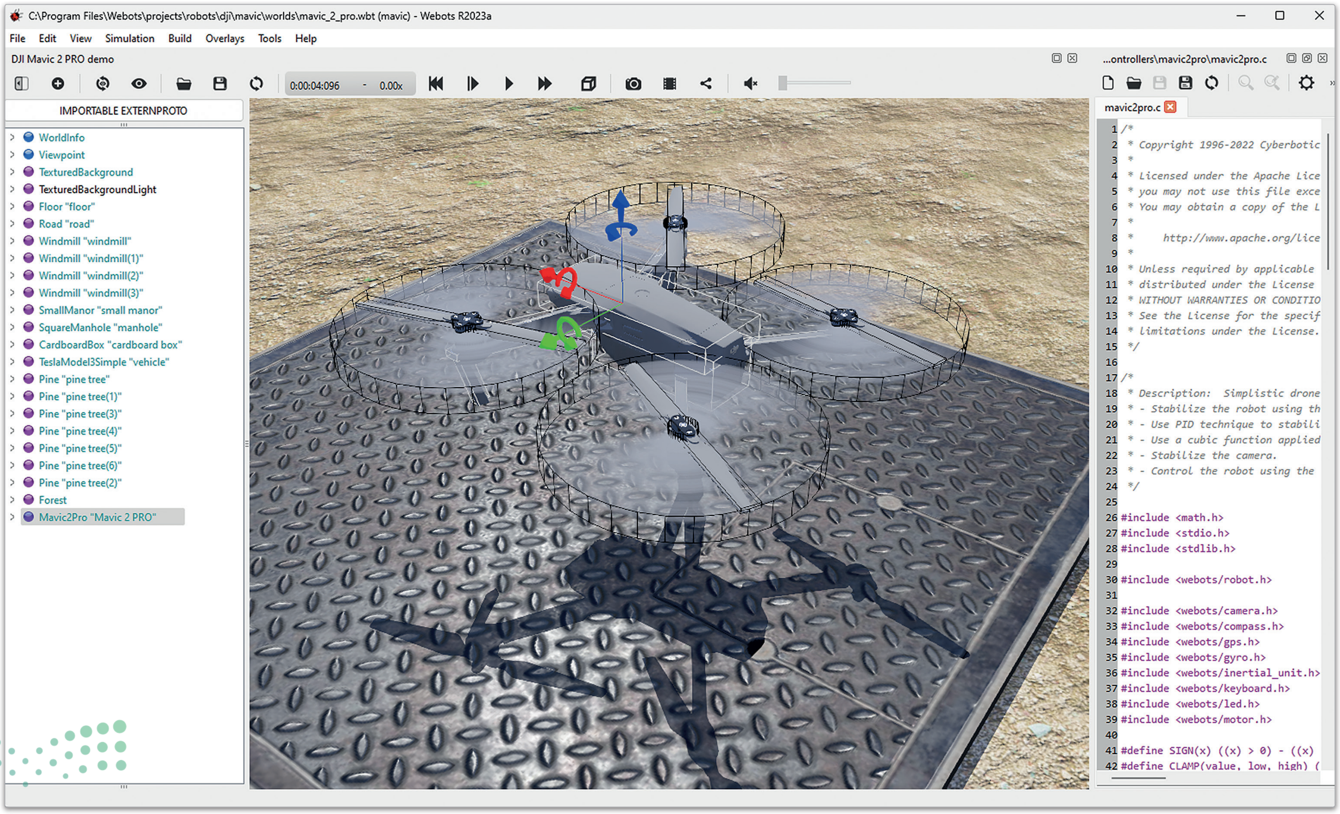
## ويبوتس Webots

ويبوتس أداة برمجية قوية يُمكن استخدامها في محاكاة الروبوتات وبيئاتها، وهي منصة ممتازة تستحق إدخالها في عَالَم الروبوتات والذكاء الاصطناعي، حيث يستطيع الطلبة تصميم الأنظمة والخوارزميات الروبوتية ومحاكاتها واختبارها باستخدام هذه الأداة، دون الحاجة إلى معدات حاسوبية باهظة الثمن.

يُعدُّ استخدام أداة ويبوتس في الذكاء الاصطناعي مفيدًا بشكل خاص؛ لأنها تتيح للطلبة تجربة خوارزميات تعلُّم الآلة واختبار أدائها في بيئة تعتمد على المُحاكاة، فمن خلال إنشاء روبوتات وبيئات افتراضية يستطيع الطلبة أن يستكشفوا إمكانيات وقيود الذكاء الاصطناعي، وأن يتعلَّموا كيفية برمجة الأنظمة الذكية التي يُمكِنها اتخاذ القرارات بناءً على بيانات الزمن الواقعي.

يُمكنك تنزيل أداة ويبوتس من الرابط التالي:

https://github.com/cyberbotics/webots/releases/download/R2023a/webots-R2023a_setup.exe




شكل 6.6: مشروع طائرة مُسيَّرة باستخدام أداة ويبوتس

## مراقبة المنطقة Area Surveillance

في هذا الدرس والدرس التالي ستستخدِم أداة ويبوتس لعمل مُحاكاة لطائرة مُسيَّرة تُحلق فوق أحد المنازل، ثم ستقوم بترقيتها لتكتشف الحدود البشرية كمُراقبة، حيث تتكون المُحاكاة من طائرة مُسيَّرة تُقلع من وضع السكون على الأرض وتبدأ في الدوران حول المنزل. وفي الدرس التالي، ستُضيف ميزة رؤية الحاسب للطائرة المُسيَّرة باستخدام الكاميرا الخاصة بها باستخدام مكتبة **أوبن سي في** (OpenCV)، وهذا سيُمكنك من تحليل الصور التي التقطتها الكاميرا.

يتم التحكم في الطائرة المُسيَّرة بواسطة نصٍّ برمجي بلغة البايثون وهو مسؤول عن التحكم في جميع الأجهزة المُسيَّرة بما فيها مُحركات المراوح والكاميرا و**نظام تحديد المواقع العالمي** (Global Positioning System - GPS) وما إلى ذلك، كما أنه يحتوي على مقطع برمجي لمزامنة جميع المُحركات لتحريك الطائرة المُسيَّرة إلى **نقاط الطريق** (Waypoints) المتنوعة وجعلها مستقرة في الهواء.

> **نقطة الطريق (Waypoint):**
> نقطة الطريق هي موقع جغرافي محدَّد في فضاء ثلاثي الأبعاد تتم برمجة الطائرة المُسيَّرة لتطير إليها أو تمر من خلالها. وتُستخدم نقاط الطريق لإنشاء مسارات طيران معرَّفة مسبقًا لتتبعها الطائرات المُسيَّرة، ويمكن ضبطها باستخدام إحداثيات نظام تحديد المواقع العالمي أو أنظمة أخرى قائمة على المواقع.

## البدء مع ويبوتس Starting with Webots

ستتعرَّف في هذا الدرس على أداة ويبوتس وبيئتها، حيث تتكون محاكاة ويبوتس من عنصرين:

- التعريف بروبوت واحد أو أكثر وبيئاتها في ملف **عَالَم ويبوتس** (Webots World).
- برنامج مُتحكِّم واحد أو أكثر للروبوتات المذكورة.

**عَالَم ويبوتس** (Webots World) هو وصف ثلاثي الأبعاد لخصائص الروبوت، حيث يتم تعريف كل كائن بما في ذلك موقعه، واتجاهه، وهندسته، ومظهره مثل: لونه أو سطوعه، وخصائصه المادية، ونوعه وما إلى ذلك، كما يُمكن أن تحتوي الكائنات على كائنات أخرى في الأنظمة الهرمية التي تُشكل العوالم. على سبيل المثال، قد يحتوي الروبوت على عجلتين، ومُستشعر مسافة، ومفصل يحتوي على كاميرا، ونحوها. يحدِّد **ملف العَالَم** (World File) فقط اسم المُتحكِّم اللازم لكل روبوت، ولا يحتوي على المقطع البرمجي **للمُتحكِّم** (Controller) في الروبوتات، وتُحفظ العَوالِم في ملفات بتنسيق ".wbt"، ويحتوي كل مشروع ويبوتس على مجلد فرعي بعنوان worlds (العَوالِم) تُخزَّن فيه الملفات بتنسيق ".wbt".

**مُتحكِّم ويبوتس** (Webots Controller) هو برنامج حاسب يتحكم في روبوت محدَّد في ملف العَالَم، ويُمكن استخدام أي لغة من لغات البرمجة التي يدعمها ويبوتس لتطوير المُتحكِّم مثل: لغة **سي بلس بلس** (C ++) ولغة **جافا** (Java)، ولكنك ستستخدم في هذا المشروع لغة البايثون. يُطلِق ويبوتس كل برنامج من برامج المُتحكِّم المُعطاة كعملية منفصلة عندما تبدأ المُحاكاة، ويقوم بربط عمليات المُتحكِّم بالروبوتات التي تمت محاكاتها، وعلى الرغم من أن العديد من الروبوتات يُمكنها مشاركة المقطع البرمجي نفسه لبرنامج المُتحكِّم، إلا أن كل روبوت سيشغِّل العملية الخاصة به. يُخزَّن مَصدر كل برنامج مُتحكِّم وملفاته الثنائية معًا في **مجلد المُتحكِّم** (Controller Directory)، حيث يحتوي كل مشروع ويبوتس على مجلد مُتحكِّم داخل المجلد الفرعي الذي يتخذ اسم controllers (المُتحكِّمات).

## بيئة الويبوتس The Webots Environment

عندما تفتح البرنامج، ستلاحظ عدة حقول ونوافذ، حيث تشمل المُكوِّنات الرئيسة لواجهة ويبوتس ما يلي:

**شريط القائمة** (Menu Bar): يقع في الجزء العلوي من الواجهة، ويُوفر شريط القوائم إمكانية الوصول إلى أوامر وخيارات متنوعة للعمل على المُحاكاة مثل: إنشاء نموذج روبوت أو استيراده، وتهيئة بيئة المُحاكاة، وتشغيل عمليات المُحاكاة.

**شريط الأدوات** (Toolbar): هو مجموعة من الأزرار الموجودة أسفل شريط القائمة ويُوفر الوصول السريع إلى الوظائف المُستخدَمة بشكل متكرر مثل: إضافة كائنات إلى المشهد، وبدء المُحاكاة وإيقافها، وتغيير عرض الكاميرا.


وزارة التعليم
Ministry of Education
2025 - 1447

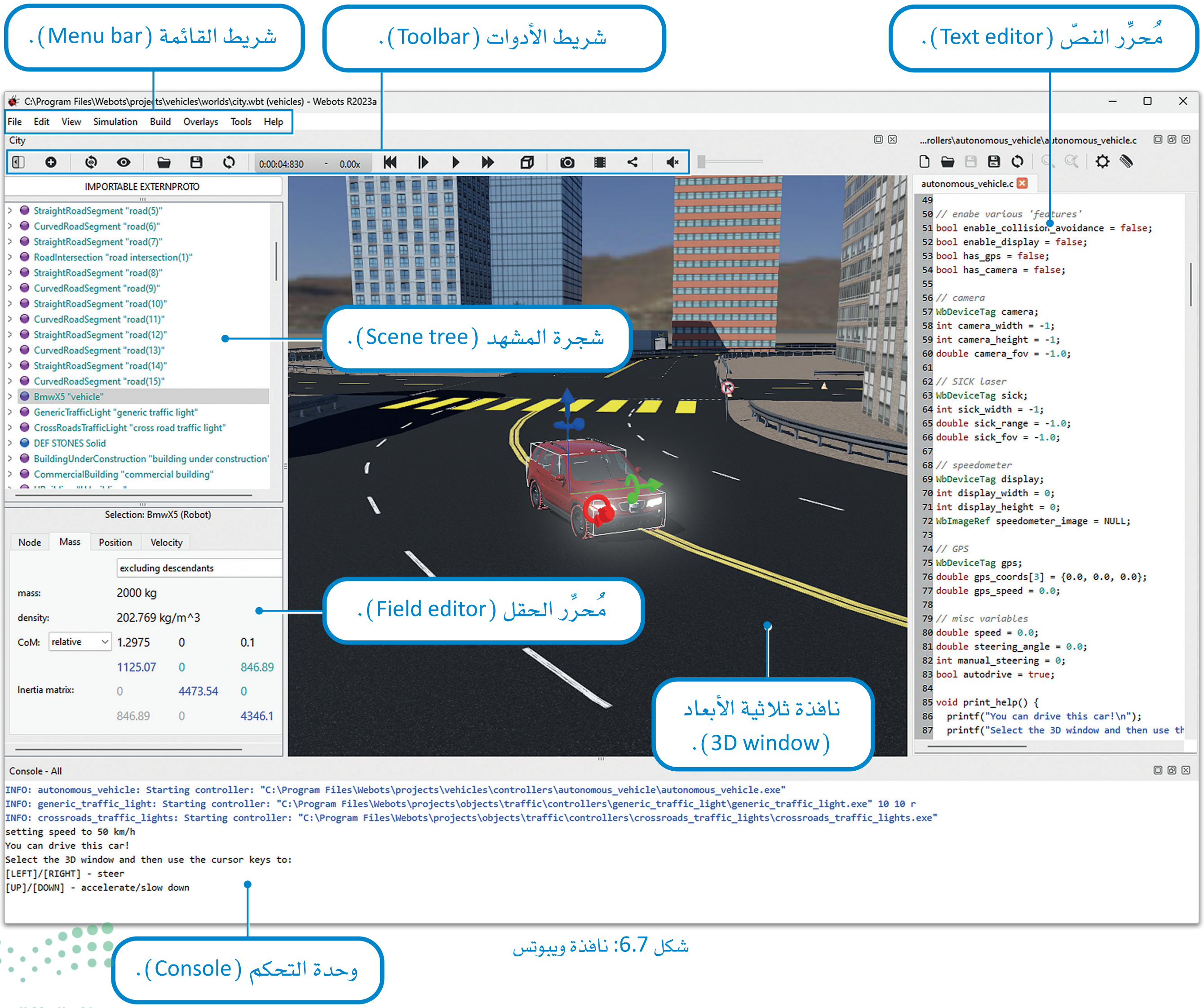
**شجرة المشهد** (Scene Tree): هي تمثيل هرمي للكائنات في بيئة المُحاكاة، حيث تتيح للمُستخدِمين التنقل في المشهد والتعامل معه مثل: إضافة أو حذف الكائنات، وتغيير خصائص الكائن، وتجميع الكائنات وإدارتها بشكل أسهل.

**مُحرِّر الحقل** (Field Editor): هو واجهة رسومات لتحرير خصائص الكائنات في بيئة المُحاكاة، حيث يُمكن للمُستخدِمين استخدامه لضبط مُعامِلات الكائن مثل: موضعه، واتجاهه، وحجمه، ومادته، وخصائصه الفيزيائية.

**نافذة ثلاثية الأبعاد** (3D Window): هي نافذة العرض الرئيس لبيئة المُحاكاة، وتعرض الكائنات وتفاعلاتها في فضاء ثلاثي الأبعاد، حيث يُمكن للمُستخدِمين التنقل في النافذة الثلاثية الأبعاد باستخدام عناصر تحكم الكاميرا المختلفة مثل: التحريك، والتكبير أو التصغير، والتدوير.

**مُحرِّر النصّ** (Text Editor): هو أداة لتحرير مصدر المقطع البرمجي أو الملفات النصّية الأخرى المُستخدَمة في المُحاكاة، ويقدِّم **تمييزًا لبناء الجمل** (Syntax Highlighting) وخصائص مفيدة أخرى لكتابة المقاطع البرمجية **وتصحيحها** (Debugging)، مثل: **الإكمال التلقائي** (Auto-Completion) و**إبراز الأخطاء** (Error Highlighting).

**وحدة التحكم** (Console): هي نافذة تعرض مُخرَجات قائمة على النصّ من المُحاكاة، بما في ذلك رسائل الخطأ ومعلومات التصحيح، وهي مفيدة في استكشاف الأخطاء التي تحدث أثناء المُحاكاة وإصلاحها.




شكل 6.7: نافذة ويبوتس


وزارة التعليم
Ministry of Education
2025 - 1447

أولًا: عليك أن تقوم بتثبيت المكتبات اللازمة التي ستستخدِمها في مشروعك. يمكنك تثبيت مكتبة أوبن سي في (OpenCV) عن طريق **باي تشارم** (PyCharm).

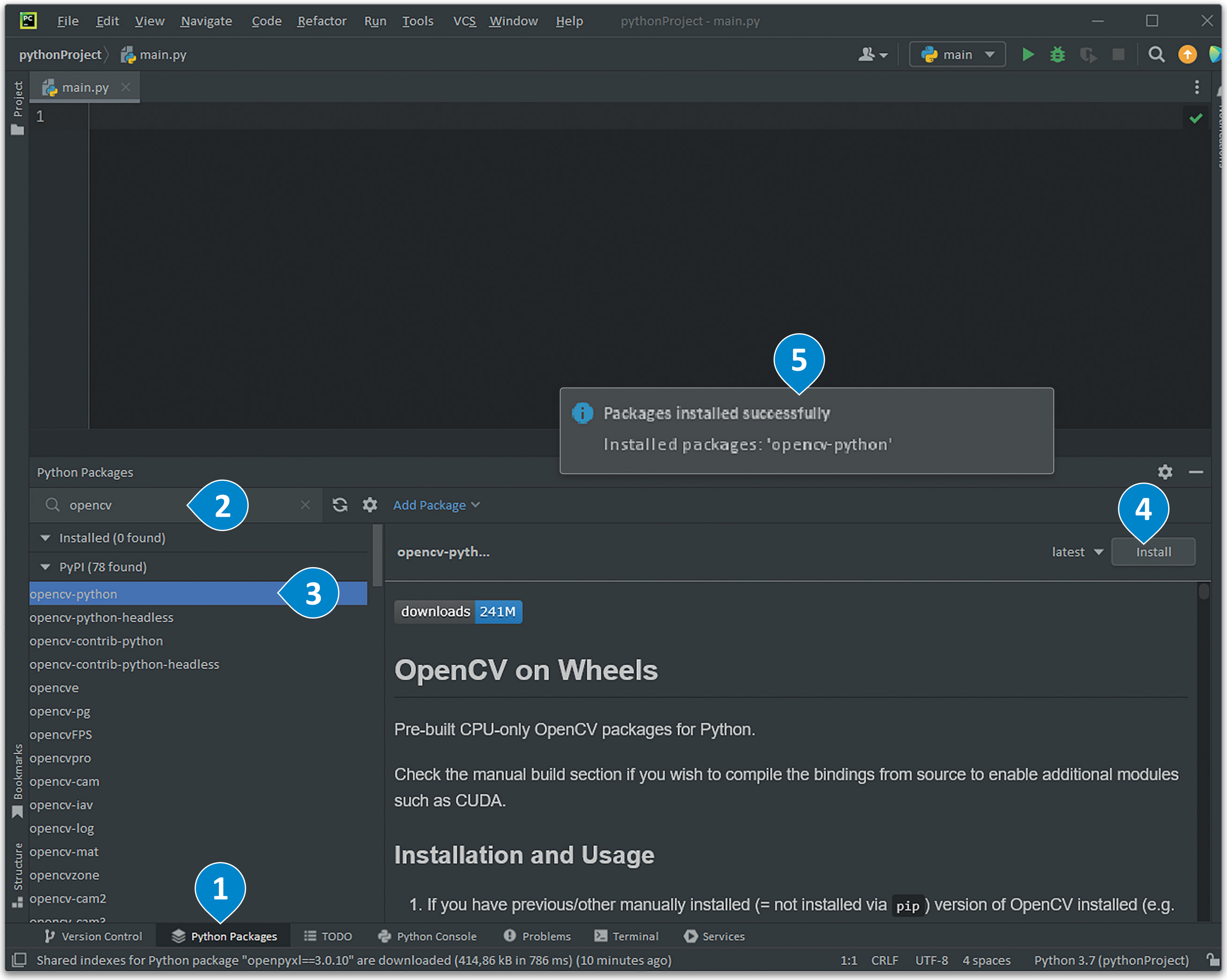
**لتنصيب مكتبة أوبن سي في (OpenCV):**

< في نافذة PyCharm (باي تشارم)، اضغط على Packages (حِزم). 1

< اكتب "opencv" (أوبن سي في) في شريط البحث. 2

< اختر opencv-python (أوبن سي في- بايثون)، 3 ثم اضغط على install (تثبيت). 4

< ستظهر لك رسالة تخبرك باكتمال التنصيب. 5




شكل 6.8: تثبيت مكتبة أوبن سي في

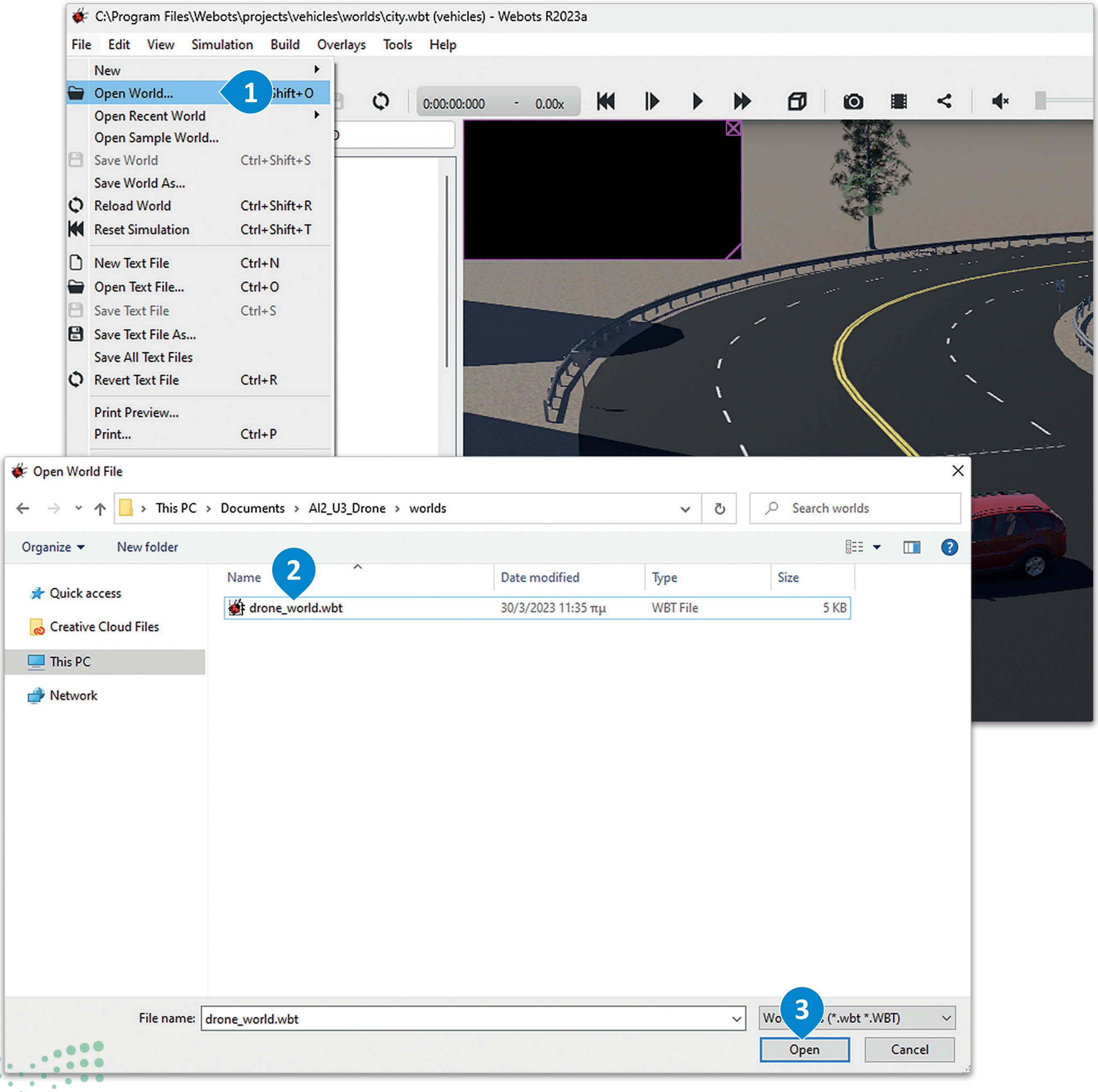
**بالمثل، يمكنك تثبيت مكتبة بيلو (Pillow) من خلال البحث عن كلمة "pillow".**

دعونا نُلقي نظرة على المشروع. أولًا: عليك أن تبحث عن ملف عَالَم ويبوتس وتقوم بتحميله.

### لفتح عَالَم ويبوتس:

> من Menu bar (شريط القائمة)، اضغط على File (ملف)، ثم على Open World (افتح عَالَم). 1

> ابحث عن ملف drone_world.wbt (الطائرة المُسيَّرة_العَالَم) في مجلد worlds (العَوالِم)، 2 ثم اضغط على Open (فتح). 3




شكل 6.9: فتح عَالَم ويبوتس

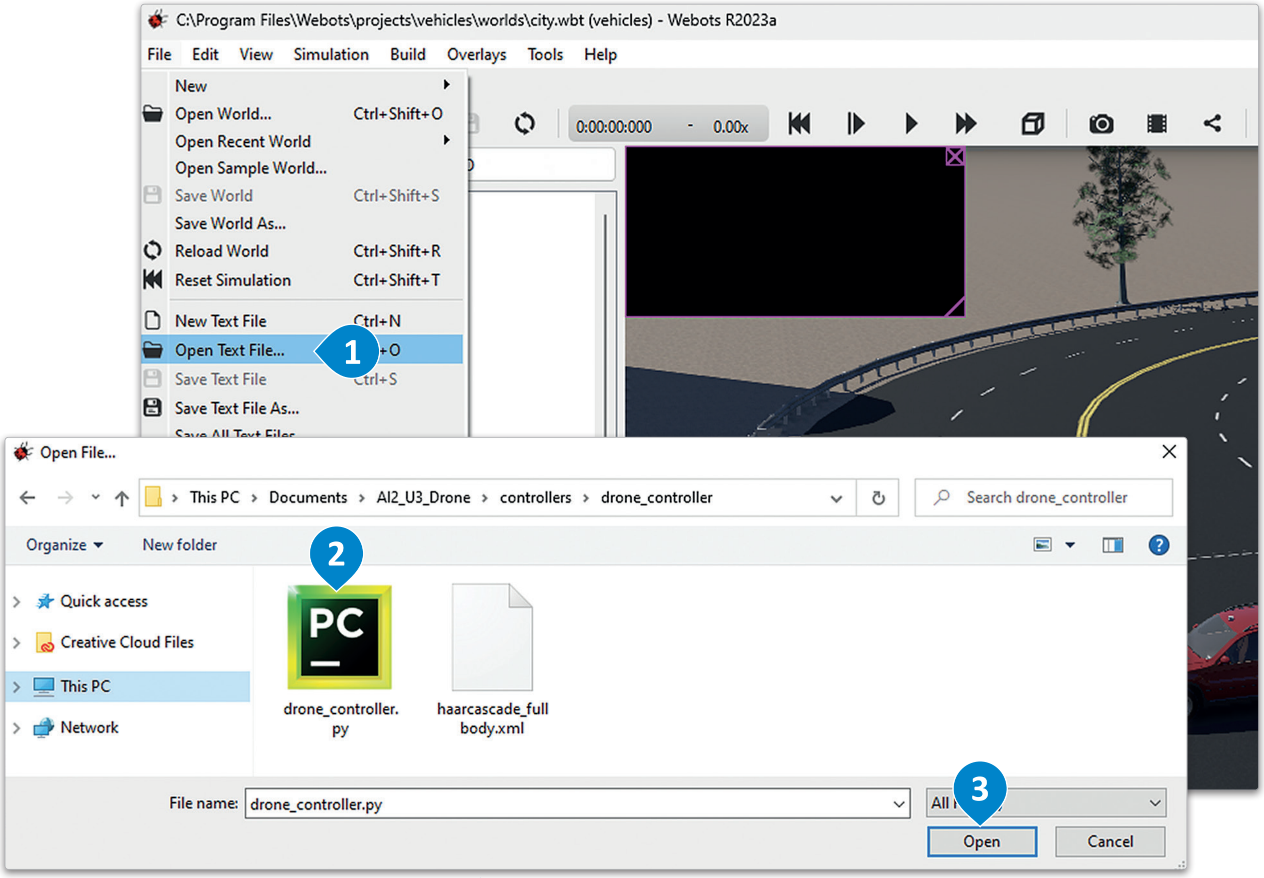

وزارة التعليم
Ministry of Education
2025 - 1447

بعدها افتح ملف النصّ البرمجي بلغة البايثون الذي سيُستخدم في التحكم في الطائرة المُسيَّرة.

**لفتح النصّ البرمجي للمُتحكِّم:**

< اضغط على File (ملف)، ثم Open Text File (افتح ملف نصّي) من شريط القائمة. 1

< ابحث عن ملف drone_controller.py (مُتحكِّم_الطائرة المُسيَّرة) في مجلد controllers (المُتحكِّمات) ثم مجلد drone_controller (مُتحكِّم_الطائرة المُسيَّرة)، 2 ثم اضغط على Open (فتح). 3




شكل 6.10: فتح النصّ البرمجي لمُتحكِّم ويبوتس

## موضع الكائن ودورانه Object Position and Rotation

تُستخدم الإحداثيات ثلاثية الأبعاد X وY وZ لتمثيل موضع كائن في الفضاء، حيث يُمثِّل X المحور الأفقي، وY المحور الرأسي، وZ محور العمق، وتُشبه إحداثيات العالَم الحقيقي لخطّ العرض وخطّ الطول والارتفاعات المُستخدَمة لوصف المواقع على الأرض.

**الانحدار** (Pitch) و**الالتفاف** (Roll) و**الانعراج** (Yaw) توجيهات دورانية يُمكن استخدامُها لوصف حركة كائن ما بالنسبة للإطار المرجعي كما يظهر في الشكل 6.11. **فالانحدار** (Pitch) هو دوران الكائن حول محوره X؛ مما يجعله يميل لأعلى أو لأسفل بالنسبة للمستوى الأفقي، أما **الالتفاف** (Roll) فهو دوران الكائن حول محوره Y؛ مما يجعل الجسم يميل جانبًا أو من جانب إلى آخر، و**الانعراج** (Yaw) هو دوران الكائن حول محوره Z؛ مما يجعل الجسم يلتف إلى اليسار أو اليمين بالنسبة للإطار المرجعي.

يُمكن استخدام هذه القيم الست معًا (X، Y، Z، الانحدار، الالتفاف، الانعراج) لوصف موضع كائن في الفضاء ثلاثي الأبعاد واتّجاهه، حيث تُستخدم بشكل شائع في الروبوتات، وأنظمة الملاحة، والتطبيقات الأخرى التي تتطلب تحديد المواقع والتحكم بدقة.

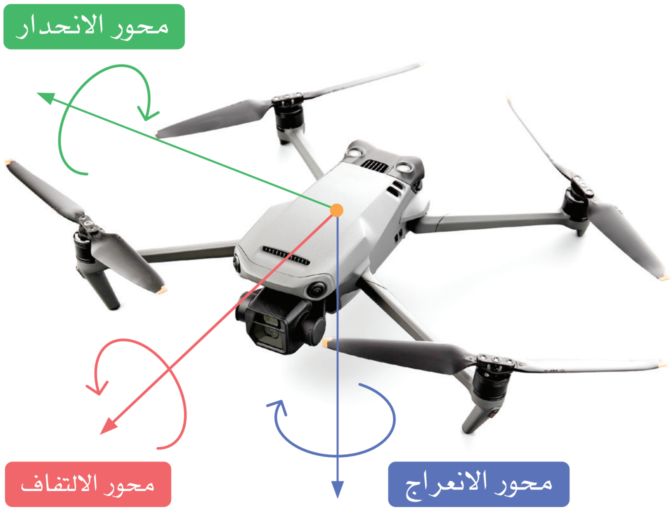
وزارة التعليم
Ministry of Education
2025 - 1447

## أجهزة الطائرة المُسيَّرة Drone Devices

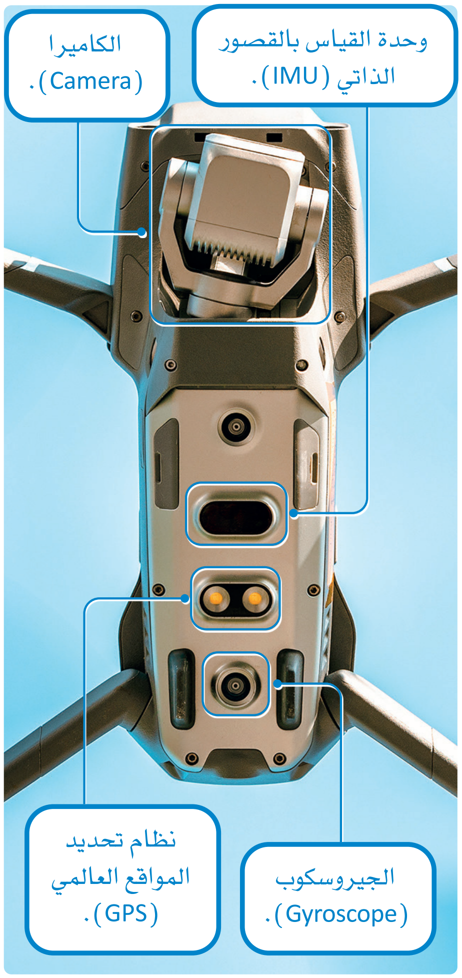
تم تجهيز **الطائرة المُسيَّرة** (Drone) بعدة **مُستشعرات** (Sensors) تتيح لها أن تجمع المُدخَلات من بيئتها، ويوفّر المُحاكي الدّالتين ()getDevice و ()enable للتفاعل مع المُستشعرات و**المُشغِّلات** (Actuators) المختلفة لروبوت المُحاكاة.

تُستخدم دالة ()getDevice للحصول على قراءات جهاز مثل: المُستشعر أو المُشغِّل من نموذج روبوت ويبوتس، وتأخذ مُعامِلًا نصّيًّا وتحدِّد اسم الجهاز المراد الوصول إليه.

تُستخدم الدالة ()enable لتنشيط جهاز، بحيث يُمكنه البدء في تقديم البيانات أو تنفيذ إجراء محدَّد.



شكل 6.11: محاور الدوران



شكل 6.12: طائرة مُسيَّرة بمُستشعرات وكاميرا

يُمكن **لوحدة القياس بالقصور الذاتي** (Inertial Measurement Unit – IMU) قياس التسارع الخطيّ للطائرة المُسيَّرة وسرعتها الزاويّة، وقياس القوى مثل الجاذبية، بالإضافة إلى قوى الدوران المؤثرة على الطائرة المُسيَّرة، كما يُمكنها أن توفر معلومات عن وضع الطائرة المُسيَّرة (الانحدار، والالتفاف، والانعراج)، وهو أمر بالغ الأهمية لتحقيق الاستقرار والتحكم.

**نظام تحديد المواقع العالَمي** (Global Positioning System – GPS) هو نظام ملاحة يعتمد على القمر الصناعي ويُوفر للطائرة المُسيَّرة معلومات دقيقة عن المواقع، ويمكِّن نظام تحديد المواقع العالَمي الطائرة المُسيَّرة من معرفة موقعها الحالي وارتفاعها وسرعتها بالنسبة إلى الأرض، وهذه المعلومات مهمة؛ للتنقل والتحكم في الطائرة المُسيَّرة.

**المُستشعرات (Sensors) هي أجهزة تكشف الكميات الفيزيائية أو الأحوال البيئية وتقيسها، وتحوّلها إلى إشارة كهربائية للمراقبة أو التحكم.**

**المُشغِّلات (Actuators) هي أجهزة تحوِّل الإشارات الكهربائية إلى حركة ميكانيكية لأداء عمل معيّن أو مُهِمَّة معيّنة.**

**بينما تقيس السرعة الخطية المسافة التي يقطعها الجسم خلال الثانية، فإنَّ السرعة الزاوية تقيس سرعة دوران الجسم حول نقطة مركزية أومحور، حيث تقيس مقدار التغيّر في الزاوية المركزية لجسم خلال وحدة الزمن، وعادةً ما تُقاس بالراديان في الثانية (rad/s) أو الدرجات في الثانية (°/s).**

**الجيروسكوب** (Gyroscope) هو مُستشعر يقيس السرعة الزاويّة، أو معدل الدوران حول محور معيّن، ويُعدُّ الجيروسكوب مفيدًا بشكل خاص في اكتشاف التغيرات الصغيرة في اتجاه الطائرة المُسيَّرة وتصحيحها، وهو أمر مهم للحفاظ على الاستقرار والتحكم أثناء الطيران.

**كاميرا الطائرة المُسيَّرة** (Drone's Camera) تُستخدم لالتقاط الصور أثناء الطيران، ويُمكن تثبيتها على الطائرة المُسيَّرة، بحيث تتمكّن من التقاط صور من جهات وزوايا مختلفة عن طريق ضبط زاوية **انحدار الكاميرا** (Camera Pitch) باستخدام الدالة ()setPosition. وفي هذا المشروع، ضُبط الموضع على 0.7، أي حوالي 45 درجة بالنظر إلى الأسفل.

أجهزة **المروحيات الأربعة** (Four Propeller) في الطائرة المُسيَّرة هي مُشغِّلات تتحكم في سرعة دوران **المروحية الرباعية** (Quadcopter) واتجاهها، وهي طائرات مُسيَّرة مُجهزة بأربعة **دوَّارات** (Rotors)، اثنان منهما يدوران في اتجاه عقارب الساعة والاثنان الآخران يدوران عكس اتجاهها، حيث يولِّد دوَران هذه الدوَّارات **قوة رفع** (Lift) ويسمح للطائرة المُسيَّرة بالإقلاع والمناورة في الهواء. وكما هو الحال مع باقي الأجهزة، تُسترد المحرِّكات وتوضع في موضعها، وتُستخدم الدالة ()setVelocity كذلك لضبط السرعة الأولية للأجهزة المروحية.

شكل 6.13: طائرة مُسيَّرة بأربع مروحيات

## التحرُّك نحو الهدف Moving to a Target

للانتقال من موقع إلى آخر، تستخدِم الطائرة المُسيَّرة دالة ()move_to_target التي تحتوي على **منطق التحكم** (Control Logic)، حيث تأخذ قائمة الإحداثيات كمُعامِل، في شكل أزواج [X، Y]؛ لاستخدامها كنقاط طريق.

في البداية، تتحقق الدالة ممّا إذا تمّت تهيئة (Initialized) **الموضع المستهدَف** (Target Position) أم لا، وفي تلك الحالة تضبطه على نقطة الطريق الأولى، ثم تتحقق مما إذا كانت الطائرة المُسيَّرة قد وصلت إلى الموضع المستهدف بالدقة المُحدَّدة في المُتغيِّر target_precision. وإذا كان الأمر كذلك، تنتقل الدالة إلى نقطة الطريق المستهدفة التالية.

ويجب حساب الزاوية بين الموضع الحالي للطائرة المُسيَّرة وموضعها المستهدف؛ لمعرفة مدى قوة الدوران التي يجب أن تكون عليه في الخطوة التالية، حيث تمت معايرة هذه القيمة وضبطها على النطاق [−π, π].

وبعد ذلك، تقوم الدالة بحساب اضطرابات الانعراج والانحدار المطلوبة لتوجيه الطائرة المُسيَّرة نحو نقطة الطريق المستهدفة وضبط زاوية انحدار الطائرة المُسيَّرة على التوالي.

## حسابات المحرِّكات Motor Calculations

أخيرًا، يجب حساب السرعة التي تضبط بها **المحرِّكات** (Motors)، وذلك بقراءة القيم المبدئية للمُستشعرات، أي قراءة: قيم الالتفاف والانحدار، والانعراج من وحدة القياس بالقصور الذاتي، ويتم الحصول على قيم مواضع X وY وZ من نظام تحديد المواقع العَالَمي، بينما يتم الحصول على قيم تسارع الالتفاف والانحدار من الجيروسكوب.

ويتم استخدام **الثوابت** (Constants) المختلفة التي تم تعريفها في المقطع البرمجي مسبقًا لإجراء الحسابات والتعديلات بالتزامن مع مُدخَلات المُستشعرات، وفي النهاية يتم ضبط **الدفع** (Thrust) الصحيح.

**معلومة**

**يمكن للمروحية أن تتحرك في أي اتجاه وأن تُحافظ على طيرانها مُستقرًا من خلال التحكّم في سرعة المروحيات الأربع واتّجاهها، فعلى سبيل المثال، عند زيادة سرعة الدوَّارين الموجودين على جانب واحد وتقليل سرعة الدوَّارين الآخرين، فإن الطائرة المُسيَّرة باستطاعتها الميلان والتحرك في اتجاه معيّن.**

```python
from controller import Robot
import numpy as np    # used for mathematic operations
import os    # used for folder creation
import cv2    # used for image manipulation and human detection
from PIL import Image    # used for image object creation
from datetime import datetime    # used for date and time

# auxiliary function used for calculations
def clamp(value, value_min, value_max):
    return min(max(value, value_min), value_max)

class Mavic (Robot):

    # constants of the drone used for flight
    # thrust for the drone to lift
    K_VERTICAL_THRUST = 68.5
    # vertical offset the drone uses as targets for stabilization
    K_VERTICAL_OFFSET = 0.6
    K_VERTICAL_P = 3.0          # P constant of the vertical PID
    K_ROLL_P = 50.0             # P constant of the roll PID
    K_PITCH_P = 30.0            # P constant of the pitch PID

    MAX_YAW_DISTURBANCE = 0.4
    MAX_PITCH_DISTURBANCE = -1
    # precision between the target position and the drone position in meters
    target_precision = 0.5

    def __init__(self):
        # initializes the drone and sets the time interval between updates of the simulation
        Robot.__init__(self)
        self.time_step = int(self.getBasicTimeStep())

        # gets and enables devices
        self.camera = self.getDevice("camera")
        self.camera.enable(self.time_step)

        self.imu = self.getDevice("inertial unit")
        self.imu.enable(self.time_step)

        self.gps = self.getDevice("gps")
        self.gps.enable(self.time_step)

        self.gyro = self.getDevice("gyro")
        self.gyro.enable(self.time_step)

        self.camera_pitch_motor = self.getDevice("camera pitch")
        self.camera_pitch_motor.setPosition(0.7)

        self.front_left_motor = self.getDevice("front left propeller")
        self.front_right_motor = self.getDevice("front right propeller")
        self.rear_left_motor = self.getDevice("rear left propeller")
        self.rear_right_motor = self.getDevice("rear right propeller")
        motors = [self.front_left_motor, self.front_right_motor,
                  self.rear_left_motor, self.rear_right_motor]
        for motor in motors: # mass initialization of the four motors
            motor.setPosition(float('inf'))
            motor.setVelocity(1)
```

تحتوي مكتبة برنامج المُتحكِّم على فئة Robot (روبوت) التي ستُستخدم طرائقها للتحكم في الطائرة المُسيَّرة.

استيراد المكتبات المطلوبة للحسابات والمعالجة.

تُستخدم الثوابت (Constants) الموجودة بشكل تجريبي لحساب الطيران والاستقرار.



وزارة التعليم
Ministry of Education
2025 - 1447

```
        self.current_pose = 6 * [0]   # X, Y, Z, yaw, pitch, roll
        self.target_position = [0, 0, 0]
        self.target_index = 0
        self.target_altitude = 0

    def move_to_target(self, waypoints):

        # Moves the drone to the given coordinates
        # Parameters:
        #    waypoints (list): list of X,Y coordinates
        # Returns:
        #    yaw_disturbance (float): yaw disturbance (negative value to go on the right)
        #    pitch_disturbance (float): pitch disturbance (negative value to go forward)

        if self.target_position[0:2] == [0, 0]:   # initialization
            self.target_position[0:2] = waypoints[0]

        # if the drone is at the position with a precision of target_precision
        if all([abs(x1 - x2) < self.target_precision for (x1, x2)
                    in zip(self.target_position, self.current_pose[0:2])]):

            self.target_index += 1
            if self.target_index > len(waypoints) - 1:
                self.target_index = 0
            self.target_position[0:2] = waypoints[self.target_index]

        # computes the angle between the current position of the drone and its target position
        # and normalizes the resulting angle to be within the range of [-pi, pi]
        self.target_position[2] = np.arctan2(
            self.target_position[1] - self.current_pose[1],
            self.target_position[0] - self.current_pose[0])
        angle_left = self.target_position[2] - self.current_pose[5]
        angle_left = (angle_left + 2 * np.pi) % (2 * np.pi)
        if (angle_left > np.pi):
            angle_left -= 2 * np.pi

        # turns the drone to the left or to the right according to the value
        # and the sign of angle_left and adjusts pitch_disturbance
        yaw_disturbance = self.MAX_YAW_DISTURBANCE * angle_left / (2 * np.pi)
        pitch_disturbance = clamp(
            np.log10(abs(angle_left)), self.MAX_PITCH_DISTURBANCE, 0.1)

        return yaw_disturbance, pitch_disturbance

    def run(self):

        # time intevals used for adjustments in order to reach the target altitude
        t1 = self.getTime()

        roll_disturbance = 0
        pitch_disturbance = 0
        yaw_disturbance = 0
```

تهيئة موضع المُسيَّرة (x، y، z) ودورانه (الالتفاف، الانحدار، الانعراج).

```python
        # specifies the patrol coordinates
        waypoints = [[-30, 20], [-60, 30], [-75, 0], [-40, -10]]
        # target altitude of the drone in meters
        self.target_altitude = 8

        while self.step(self.time_step) != -1:

            # reads sensors
            roll, pitch, yaw = self.imu.getRollPitchYaw()
            x_pos, y_pos, altitude = self.gps.getValues()
            roll_acceleration, pitch_acceleration, _ = self.gyro.getValues()
            self.current_pose = [x_pos, y_pos, altitude, roll, pitch, yaw]

            if altitude > self.target_altitude - 1:
                # as soon as it reaches the target altitude,
                # computes the disturbances to go to the given waypoints
                if self.getTime() - t1 > 0.1:
                    yaw_disturbance, pitch_disturbance = self.move_to_target(
                        waypoints)
                    t1 = self.getTime()

            # calculates the desired input values for roll, pitch, yaw,
            # and altitude using various constants and disturbance values
            roll_input = self.K_ROLL_P * clamp(roll, -1, 1) +
                         roll_acceleration + roll_disturbance
            pitch_input = self.K_PITCH_P * clamp(pitch, -1, 1) +
                          pitch_acceleration + pitch_disturbance
            yaw_input = yaw_disturbance
            clamped_difference_altitude = clamp(self.target_altitude -
                                altitude + self.K_VERTICAL_OFFSET, -1, 1)
            vertical_input = self.K_VERTICAL_P *
                             pow(clamped_difference_altitude, 3.0)

            # calculates the motors' input values based on the
            # desired roll, pitch, yaw, and altitude values
            front_left_motor_input = self.K_VERTICAL_THRUST + vertical_input
                                   - yaw_input + pitch_input - roll_input
            front_right_motor_input = self.K_VERTICAL_THRUST + vertical_input
                                    + yaw_input + pitch_input + roll_input
            rear_left_motor_input = self.K_VERTICAL_THRUST + vertical_input
                                  + yaw_input - pitch_input - roll_input
            rear_right_motor_input = self.K_VERTICAL_THRUST + vertical_input
                                   - yaw_input - pitch_input + roll_input

            # sets the velocity of each motor based on the motors' input values calculated above
            self.front_left_motor.setVelocity(front_left_motor_input)
            self.front_right_motor.setVelocity(-front_right_motor_input)
            self.rear_left_motor.setVelocity(-rear_left_motor_input)
            self.rear_right_motor.setVelocity(rear_right_motor_input)


robot = Mavic()
robot.run()
```

waypoints (نقاط الطريق) الخاصة بالمسار الذي ستطير فيه الطائرة المُسيَّرة.


وزارة التعليم
Ministry of Education
2025 - 1447

حان الوقت الآن لإدراج النصّ البرمجي في الطائرة المُسيَّرة وتشغيل المُحاكاة.

## لإدراج برنامج المُتحكّم وتشغيل المُحاكاة:

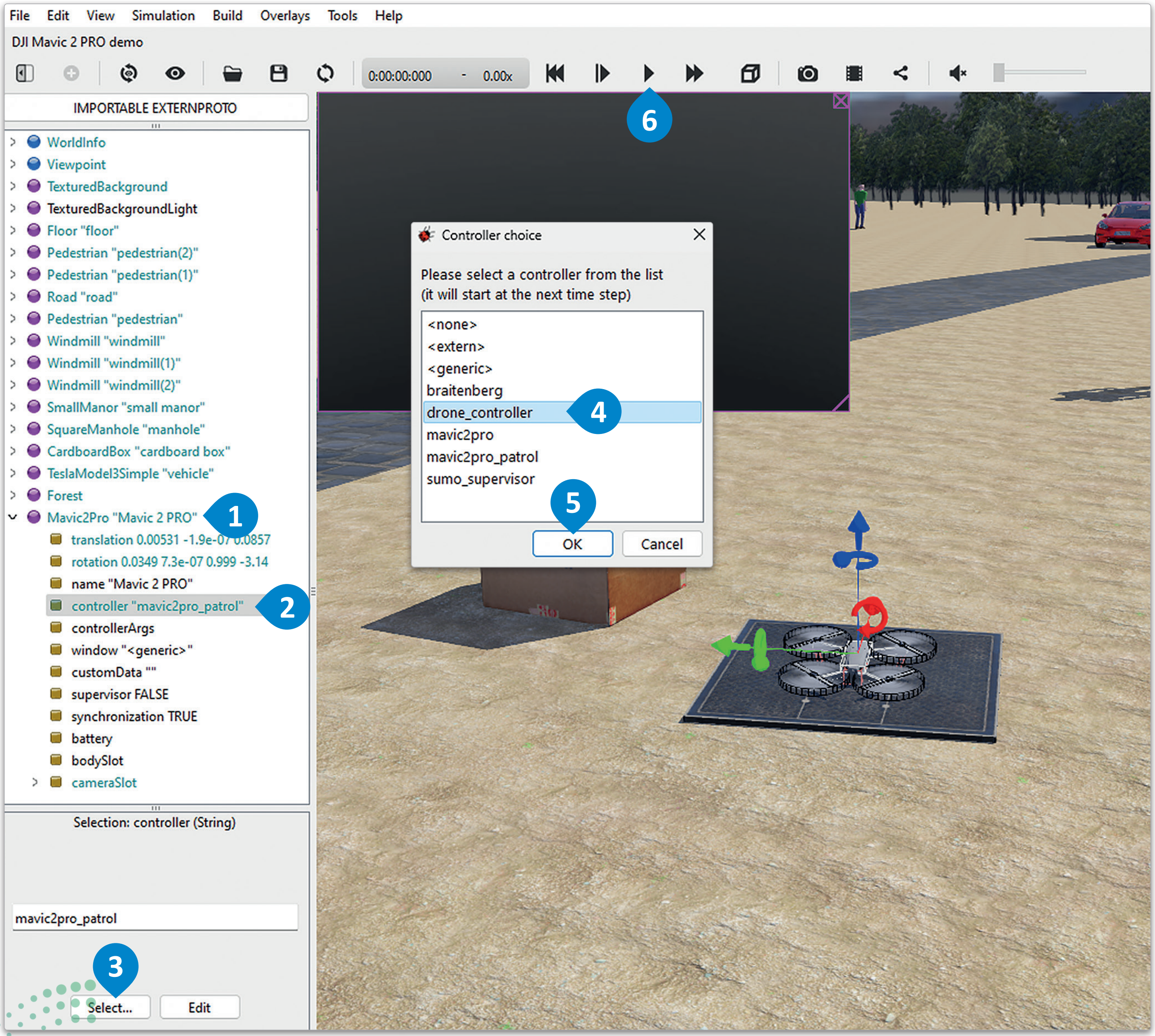
> من Scene tree (شجرة المشهد)، اضغط على "Mavic2Pro "Mavic 2 Pro، ثم اضغط على "controller "mavic2pro_patrol. 1 2

> من Field editor (مُحرِّر الحقل)، اضغط على ... Select (اختيار). 3

> حدِّد drone_controller (مُتحكِّم_الطائرة المُسيَّرة)، 4 ثم اضغط على OK (موافق). 5

> من Toolbar (شريط الأدوات)، اضغط على Run the simulation in real-time (شغّل المحاكاة بشكل فوري). 6

عند إجراء تغييرات على النصوص البرمجية، لا تنسَ أن تضغط على Ctrl + S.




شكل 6.14: إدراج النصّ البرمجي لبرنامج المُتحكِّم وتشغيل المُحاكاة

عندما تبدأ المُحاكاة، ستعمل محركات الطائرة المُسيَّرة وستُقلع، ثم ستتبع الطريق المحدَّدة مسبقًا حول المنزل، وتمر عبر نقاط الطريق.

شكل 6.15: إقلاع الطائرة المُسيَّرة

# تمرينات

1 حلّل الدالة move_to_target( ) واشرح كيفية قيام الطائرة المُسيَّرة بحساب موضعها التالي في قائمة نقاط الطريق. كيف يمكن تحسين مسار الطائرة المُسيَّرة لتقليل زمن الطيران بين نقاط الطريق؟

2 قيِّم عيوب خوارزميّة التحكُّم الحالية في الطائرة المُسيَّرة عند مواجهة عوامل خارجية مثل: الرياح أو العوائق أو عدم دقة نظام تحديد المواقع العالمي، ثم اقترح وناقش التحسينات التي يمكن القيام بها في خوارزميّة التحكم لجعل الطائرة المُسيَّرة أكثر صمودًا في وجه هذه التحديات.

وزارة التعليم
Ministry of Education
2025 - 1447

3 استكشف الآثار الأخلاقية للطائرات المُسيَّرة الهوائية في التطبيقات الواقعية مثل: المراقبة وتوصيل الطرود وعمليات البحث والإنقاذ، ثم اكتب عن المخاوف المحتملة الخاصة بالخصوصية، وقضايا السلامة، واحتمالات إساءة استخدام هذه التقنية.

4 أضف خاصية تُسجِّل موضع الطائرة المُسيَّرة وارتفاعها واتجاهها على فترات منتظمة أثناء الطيران، ثم اكتب كل الأنماط التي قد تجدها في بيانات السجل.

5 جرِّب استخدام قيم مختلفة لثوابت PID في برنامج المُتحكِّم (K_VERTICAL_P، K_ROLL_P، K_PITCH_P). ولاحظ كيفية تأثير هذا التغيرات على استقرار الطائرة المُسيَّرة واستجابتها، ثم ناقش الموازنات بين الاستقرار والاستجابة.


وزارة التعليم
Ministry of Education
2025 - 1447

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الثالث

# التطبيقات الروبوتية 2

## الروبوتية ورؤية الحاسب والذكاء الاصطناعي

## Robotics, Computer Vision and AI

**رؤية الحاسب** (Computer Vision) و**الروبوتية** (Robotics) مجالان متطوران من مجالات التقنية يعملان معًا على متابعة التغيير السريع لطريقة حياة الناس وعملهم، وعندما يُدمجان فإنهما يفتحان مجموعة واسعة من الإمكانيات **للأتمتة** (Automation) والتصنيع وتطوير التطبيقات الأخرى.

يُعدُّ الذكاء الاصطناعي مُكوِّنًا رئيسًا من مُكوِّنات رؤية الحاسب والروبوتية على حدّ سواء؛ مما يُمكِّن الآلات من التعلُّم والتكيُّف مع بيئتها بمرور الوقت، حيث تستطيع الروبوتات باستخدام خوارزميات الذكاء الاصطناعي أن تُحلِّل وتُفسِّر كميات هائلة من البيانات المرئية؛ مما يسمح لها باتخاذ قرارات والقيام بإجراءات في الوقت الفعلي. كما يُمكِّن الذكاء الاصطناعي الروبوتات من تحسين أدائها ودقتها بمرور الوقت، إذ أنها تتعلّم من تجاربها وتُعدِّل سلوكها وفقًا لذلك، وهذا يعني أن الروبوتات المزودة برؤية الحاسب وقدرات الذكاء الاصطناعي يُمكنها أداء مهام شديدة التعقيد بشكل أكثر دقّة وكفاءة.

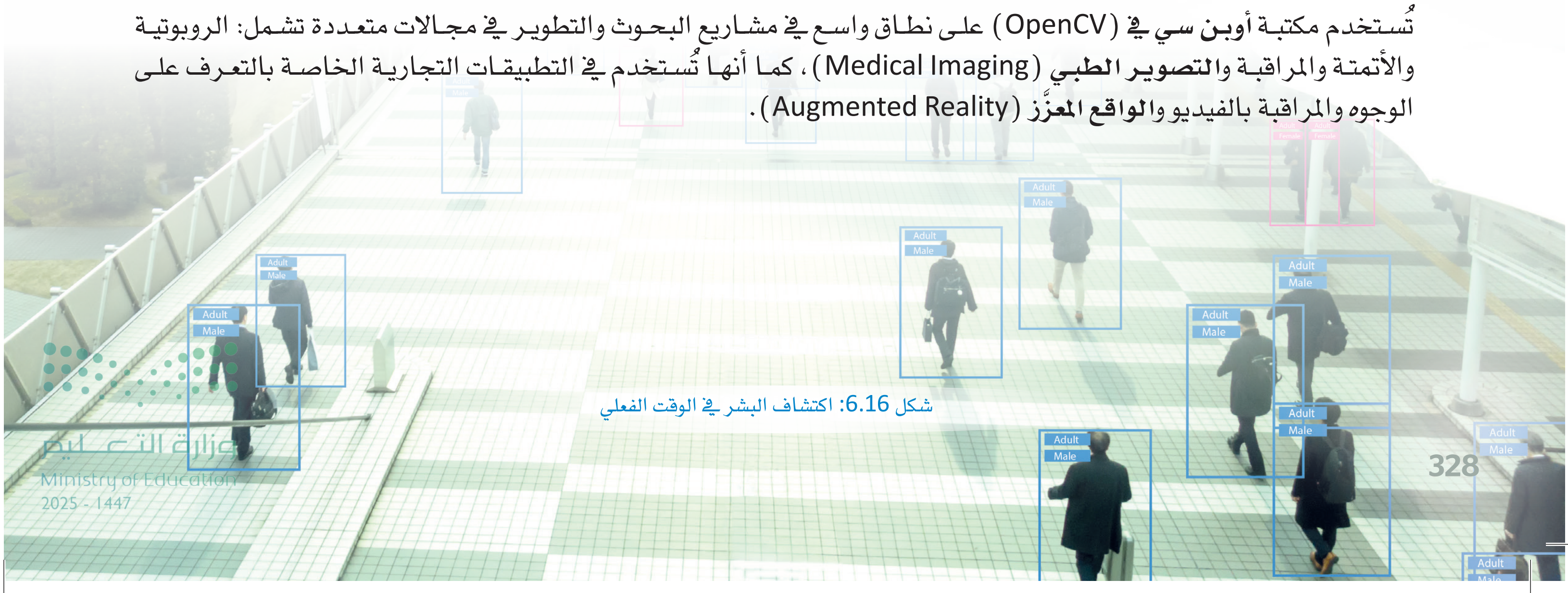
في هذا الدرس ستعمل على ترقية المشروع الأوليّ للطائرة المُسيَّرة الذي تم توضيحه في الدرس السابق، وذلك باستخدام رؤية الحاسب لاكتشاف وتحديد الشّخوص البشرية القريبة من المنزل، حيث يُمكن النظر إليهم على أنهم أعداء في سيناريو العَالَم الواقعي، وتَستخدِم الطائرة المُسيَّرة الكاميرا المزود بها؛ لتكون بمثابة نظام مراقبة، كما يُمكن تطبيق هذا المثال وتنفيذه بسهولة على العديد من المباني الأخرى والبنية التحتية والممتلكات الخاصة والشركات مثل: المصانع ومحطات توليد الطاقة.

سيتم استخدام مكتبة **أوبن سي في** (OpenCV) من لغة البايثون لاكتشاف الشّخوص البشرية، وهي مكتبة رؤية حاسوبية مفتوحة المصدر توفر مجموعة من خوارزميات رؤية الحاسب ومعالجة الصور بالإضافة إلى مجموعة من أدوات البرمجة؛ لتطوير التطبيقات في هذه المجالات.

OpenCV

يُمكن استخدام مكتبة **أوبن سي في** (OpenCV) في الروبوتية للقيام بمهام مثل: اكتشاف الكائنات وتتبُّعها، وإعادة البناء ثلاثي الأبعاد، والملاحة، وتشمل ميزاتها كذلك اكتشاف الكائنات والتعرف عليها، واكتشاف الوجوه والتعرف عليها، ومعالجة الصور ومقاطع الفيديو، و**معايرة الكاميرا** (Camera Calibration)، وتعلُّم الآلة، وغيرها.

تُستخدم مكتبة **أوبن سي في** (OpenCV) على نطاق واسع في مشاريع البحوث والتطوير في مجالات متعددة تشمل: الروبوتية والأتمتة والمراقبة و**التصوير الطبي** (Medical Imaging)، كما أنها تُستخدم في التطبيقات التجارية الخاصة بالتعرف على الوجوه والمراقبة بالفيديو و**الواقع المعزَّز** (Augmented Reality).



شكل 6.16: اكتشاف البشر في الوقت الفعلي

وزارة التعليم
Ministry of Education
2025 - 1447

لنستعرض التغييرات التي ستُجريها لإضافة وظائف رؤية الحاسب للطائرة المُسيَّرة.

## إضافة المؤقِّت Adding a Timer

يُمكن أن يكون التقاط صورة ومعالجتها وحفظها مكلفًا من الناحية الحاسوبية إذا حُسب لكل إطار من إطارات المُحاكاة، ولذلك ستضيف مؤقِّتًا زمنيًا لاستخدامه؛ لتنفيذ هذه الإجراءات كل خمسِ ثوانٍ فقط.

```
# time intervals used for adjustments in order to reach the target altitude
t1 = self.getTime()
# time intervals between each detection for human figures
t2 = self.getTime()
```

## إنشاء مجلد Creating a Folder

سيتم حفظ الصور المُلتَقطة التي يتم فيها اكتشاف الشّخوص البشرية في مجلد، حيث يُعدّ جزءًا من أرشيف المراقبة الأمنية الذي سيساعد على فحص الصور في المستقبل.

أولًا: عليك أن تستخدِم الدالة getcwd( ) لتسترد مسار دليل العمل الحالي لبرنامج المُتحكِّم (وهو المجلد الذي يتضمّن برنامج المُتحكِّم) حتى يتعرف البرنامج على المكان الذي يضع فيه المجلد الجديد باسم: detected (تم الاكتشاف)، بحيث تُستخدم الدالة path.join( ) لربط اسم المسار بسلسلة اسم المجلد النصّية، وتتمثّل الخطوة الأخيرة في التحقق مما إذا كان المجلد موجودًا بالفعل أم لا، وفي تلك الحالة يتم إنشاء مجلد جديد.

```
# gets the current working directory
cwd = os.getcwd()
# sets the name of the folder where the images
# with detected humans will be stored
folder_name = "detected"
# joins the current working directory and the new folder name
folder_path = os.path.join(cwd, folder_name)

if not os.path.exists(folder_path):
# creates the folder if it doesn't exist already
    os.makedirs(folder_path)
    print(f"Folder \"detected\" created!")
else:
    print(f"Folder \"detected\" already exists!")
```

## معالجة الصورة Image Processing

في هذا التوقيت يمكنك الآن استرداد (قراءة) الصورة من الجهاز لمعالجتها قبل محاولة الكشف. لاحظ أن كل ما يتعلق بمعالجة الصورة وصولًا إلى حفظها يحدث كل خمسِ ثوانٍ فقط، كما هو مبيَّن في الشرط "self.getTime( ) – t2> 5.0".

```
# initiates the image processing and detection routine every 5 seconds
if self.getTime() - t2 > 5.0:

    # retrieves image array from camera
    cameraImg = self.camera.getImageArray()
```

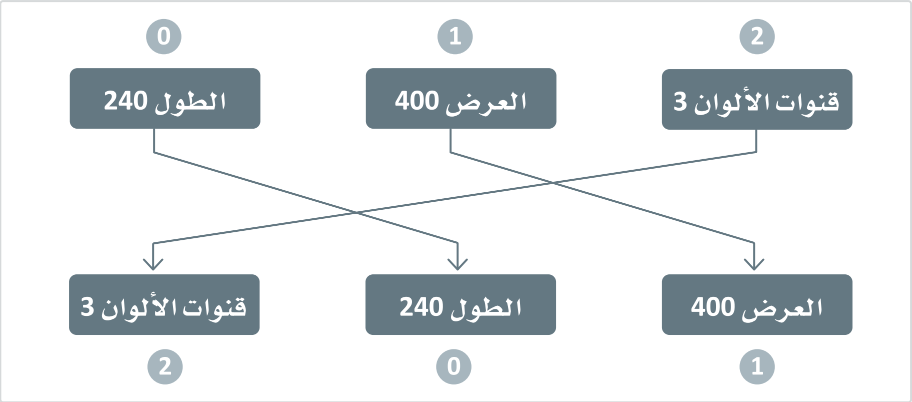

وزارة التعليم
Ministry of Education
2025 - 1447

بعد التحقق من استرداد الصورة بنجاح، تنتقل الخوارزميّة إلى تعديل بعض خصائصها، بحيث تكون الصورة ثلاثية الأبعاد، ولها أبعاد طول وعرض وقنوات ألوان، حيث تلتقط كاميرا الطائرة المُسيَّرة صورًا بارتفاع 240 بكسل وعرض 400 بكسل، كما أنها تستخدِم 3 قنوات ألوان لحفظ معلومات الصورة وهي: الأحمر والأخضر والأزرق.

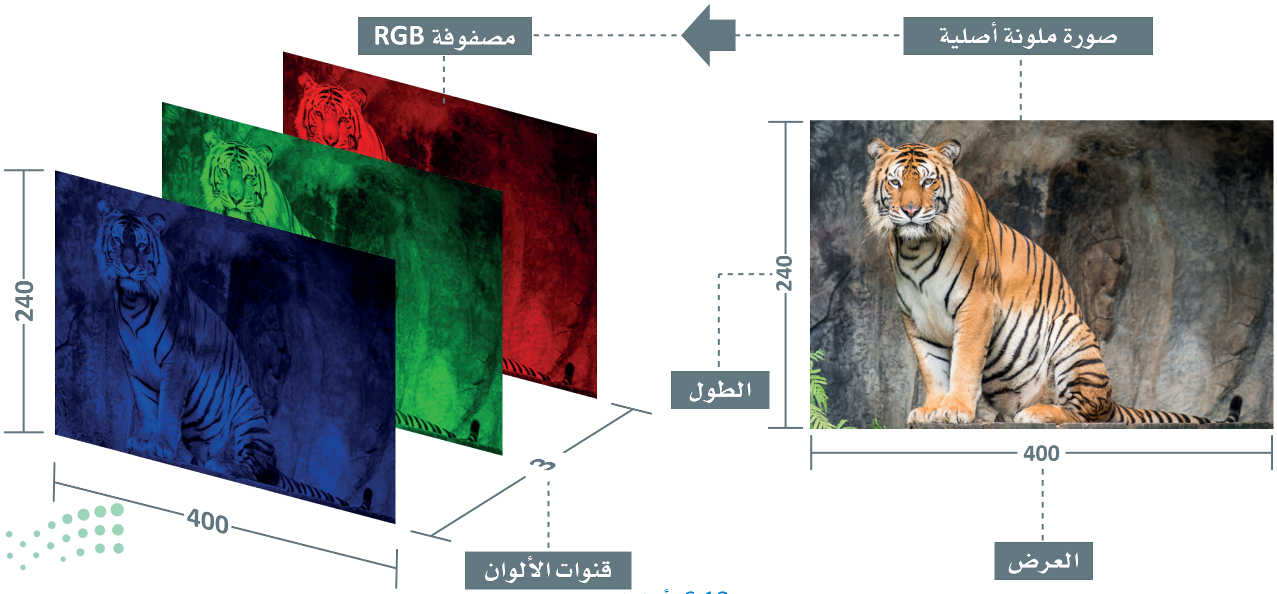
يجب معالجة الصورة أولًا حتى يتم استخدامها في الكشف، ولكي يتم تطبيق الدوال بشكل صحيح في وقت لاحق، لا بُدّ أن تحقق الصورة تركيبًا معيّنًا. في هذا المثال، يجب أن يتغير تسلسل الأبعاد من (الطول، والعرض، وقنوات الألوان) إلى (قنوات الألوان، والطول، والعرض) باستخدام الدالة ()transpose، حيث تُقدَّم **صورة الكاميرا** (CameraImg)، والتسلسل الجديد (1، 0، 2) كمُعامِلات لهذه الدالة، بافتراض أن الترتيب الأصلي كان (2، 1، 0).

كما يجب تعديل أحجام الأبعاد بعد تغيير التسلسل، حيث تُستخدم الدالة ()reshape بالطريقة نفسها، ولكن أحجام الأبعاد المعنيّة كالمُعامِل الثاني منها تكون (400، 240، 3).

```
# reshapes image array to (channels, height, width) format
cameraImg = np.transpose(cameraImg, (2, 0, 1))
cameraImg = np.reshape(cameraImg, (3, 240, 400))
```



شكل 6.17: تغيير تسلسل الأبعاد



شكل 6.18: أبعاد الصورة

بعد ذلك، يجب تغيير الصورة إلى التدرج الرمادي حيث أن خوارزميّة الاكتشاف تستلزم ذلك، مع وجوب تخزينها أولًا في كائن صورة ووجوب الجمع بين قنوات ألوانها الثلاثة، وهنا يجب دمج قنوات الألوان وتخزينها باستخدام الدالة ( )merge في تسلسل عكسي: أي أن يكون تسلسل الألوان (أزرق، أخضر، أحمر) بدلًا من (أحمر، أخضر، أزرق)، وأن يكون تسلسلها الرقمي (0، 1، 2) بدلًا من (2، 1، 0) على الترتيب.

```
# creates RGB image from merged channels
img = Image.new('RGB', (400, 240))
img = cv2.merge((cameraImg[2], cameraImg[1], cameraImg[0]))
```

وأخيرًا، يتم تحويل الصورة إلى التدرج الرمادي باستخدام الدالة ( )cvtColor التي تستخدِم مُعامِل COLOR_BGR2GRAY لتغيير الألوان من الأزرق والأخضر والأحمر إلى التدرج الرمادي.

```
# converts image to grayscale
gray = cv2.cvtColor(np.uint8(img), cv2.COLOR_BGR2GRAY)
```

## اكتشاف صور الحدود البشرية Human Silhouette Detection

لكي تكتشف الصورة، عليك أن تستخدم **مصنِّف هار كاسكيد** (Haar Cascade Classifier)، وهو خوارزميّة لاكتشاف الكائنات تعتمد على تعلُّم الآلة، وتُستخدم لتحديد الكائنات في الصور أو مقاطع الفيديو. ولاستخدام هذا المُصنِّف تحتاج أن تُدرِّب نموذج تعلُّم الآلة على مجموعة من الصور التي تحتوي على الكائن الذي تريد البحث عنه، وعلى صور أخرى لا تحتوي على هذا الكائن، حيث تقوم الخوارزميّة بالبحث عن أنماط معيّنة في الصور لتحديد مكان الكائن. وفي العادة تُستخدم هذه الخوارزميّة للعثور على أشياء محدَّدة مثل: الوجوه، أو أشخاص يسيرون في مقطع فيديو. ومع ذلك قد لا تعمل هذه الخوارزميّة بشكل جيد في بعض المواقف التي يكون فيها الكائن محجوبًا جزئيًّا أو كليًّا أو معرضًا لإضاءة منخفضة. تم تدريب المصنِّف في مشروعك تدريبًا خاصًّا على اكتشاف البشر، وعليك أن تستخدم ملف haarcascade_fullbody.xml الذي ستُزود به، وهو نموذج تعلُّم آلة مُدرَّب مسبقًا ويشكِّل جزءًا من مكتبة **أوبن سي في** (OpenCV)، ويُقدَّم كمُعامِل لكائن ( )CascadeClassifier، ثم تَستخدم الدالة ( )detectMultiScale للقيام بعملية الاكتشاف.

```
# loads and applies the Haar cascade classifier to detect humans in image
human_cascade = cv2.CascadeClassifier('haarcascade_fullbody.xml')
humans = human_cascade.detectMultiScale(gray)
```

شكل 6.19: مثال على إكتشاف صور الحدود البشرية

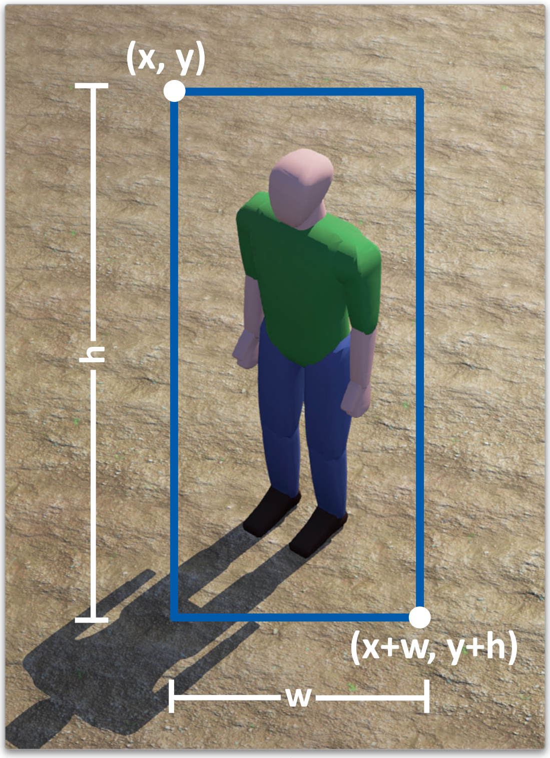

وزارة التعليم
Ministry of Education
2025 - 1447

## تقرير الطائرة المُسيَّرة وحفظ الصور المُكتشَفة

## Drone Report and Saving of the Detected Images

الإضافة النهائية لبرنامج المُتحكِّم الخاص بك هو نظام تقرير بسيط تُقدِّمه الطائرة المُسيَّرة عن طريق طباعة رسالة على **وحدة التحكم** (Console) عند اكتشاف شكل بشري، وحفظ الصورة في المجلد الذي أنشأتَه من قبل.

يقوم المُتغيِّر humans (البشر) بحمل المستطيلات الإطارية التي يُكتشف البشر بداخلها في حال عُثر عليهم. تُعرَّف المستطيلات بواسطة أربعة مُتغيِّرات: وهي الزوج x و y اللذان يمثِّلان الإحداثيين اللذين في الصورة وذلك في الزاوية العُليا من الجهة اليُسرى للمستطيل، وكذلك الزوج w و h، الذي يمثِّل عرض المستطيل وارتفاعه. في جميع الاكتشافات الموجودة في الصورة تُحدِّد الدالة rectangle() البشر بمستطيل أزرق، حيث تنظر الدالة إلى مُتغيِّرات الصورة على أنها تتمثّل في الزاوية اليسرى العُلوية (x، y) والزاوية اليمنى السُفلية (x+w، y+h) من المستطيل، ولون المستطيل وعرضه، وفي الصورة الموضَّحة تلاحظ أن لون المستطيل أزرق (B=255، G=0، R=0) وعرضه 2.




شكل 6.20: مُتغيِّرات المستطيل

سيقوم نظام التقرير باسترجاع التاريخ والوقت الحاليين باستخدام الدالة datetime.now() وطباعتها على وحدة التحكم، بالإضافة إلى إحداثيات الطائرة المُسيَّرة في وقت التقرير، ويتم تعديل تنسيق التاريخ والوقت بطريقة بسيطة عن طريق إدراج الشرطات العُلوية (-) والشرطات السُفلية (_) لاستخدامها كجزء من اسم الملف المحفوظ، ثم يتم حفظها في المجلد باستخدام الدالة imwrite()، وعند اكتمال كل شيء تقوم الدالة getTime() بإعادة ضبط المؤقِّت.

```
# loop, through detected human images, annotates them with a bounding box
# and prints a timestamp and an info message on the console
for (x, y, w, h) in humans:

    # the image, the top left corner, the bottom right corner, color and width of the rectangle
    cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)
    current_time = datetime.now()
    print(current_time)
    print("Found a person in coordinates [{:.2f}, {:.2f}]"
                    .format(x_pos, y_pos))

    # saves annotated image to file with timestamp
    current_time = current_time.strftime("%Y-%m-%d_%H-%M-%S")
    filename = f"detected/IMAGE_{current_time}.png"
    cv2.imwrite(filename, img)

t2 = self.getTime()
```

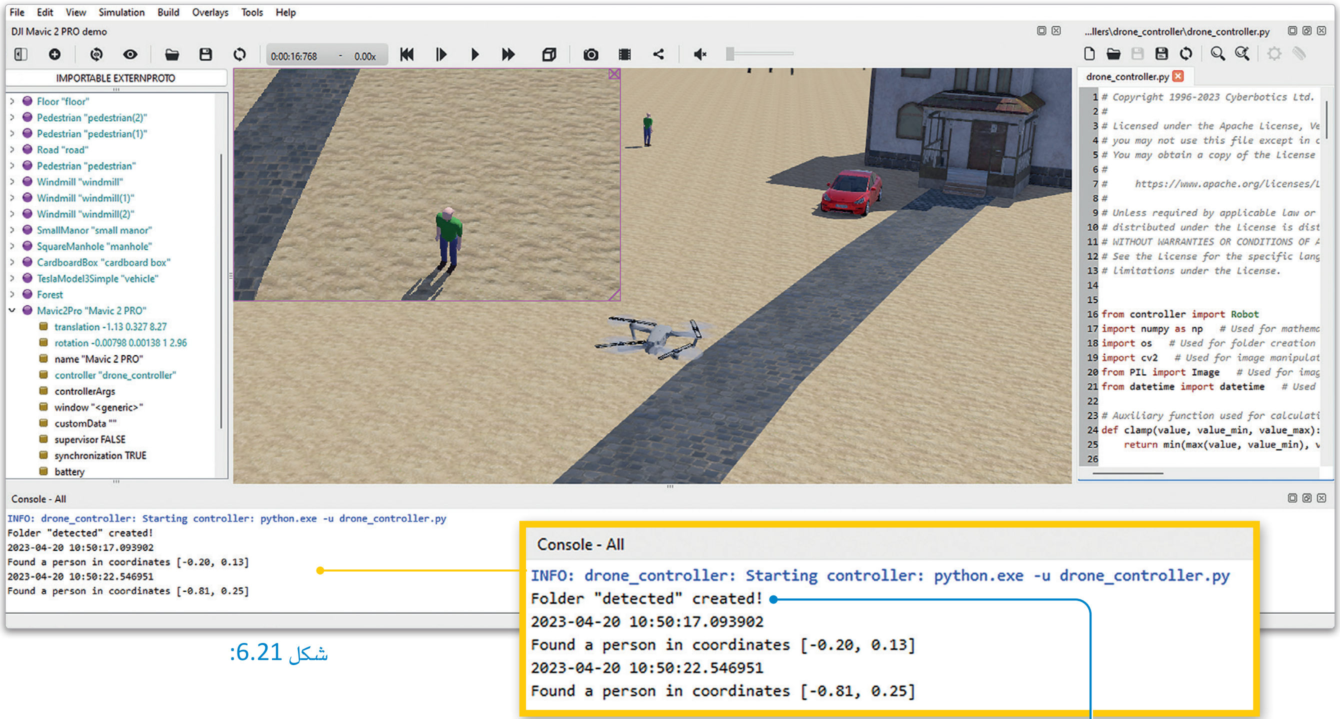
في السلسلة النصّية، يتم استخدام الترميز {:.2f} كاختصار **لعدد حقيقي** (floating number) ذي خانتين عشريتين، وهنا يتم استخدام الاختصارين للمُتغيِّرين x_pos و y_pos.

بعد إضافة كل هذه الوظائف يجب أن تظهر الدالة ( )run الخاصة ببرنامج المُتحكّم كما يلي:

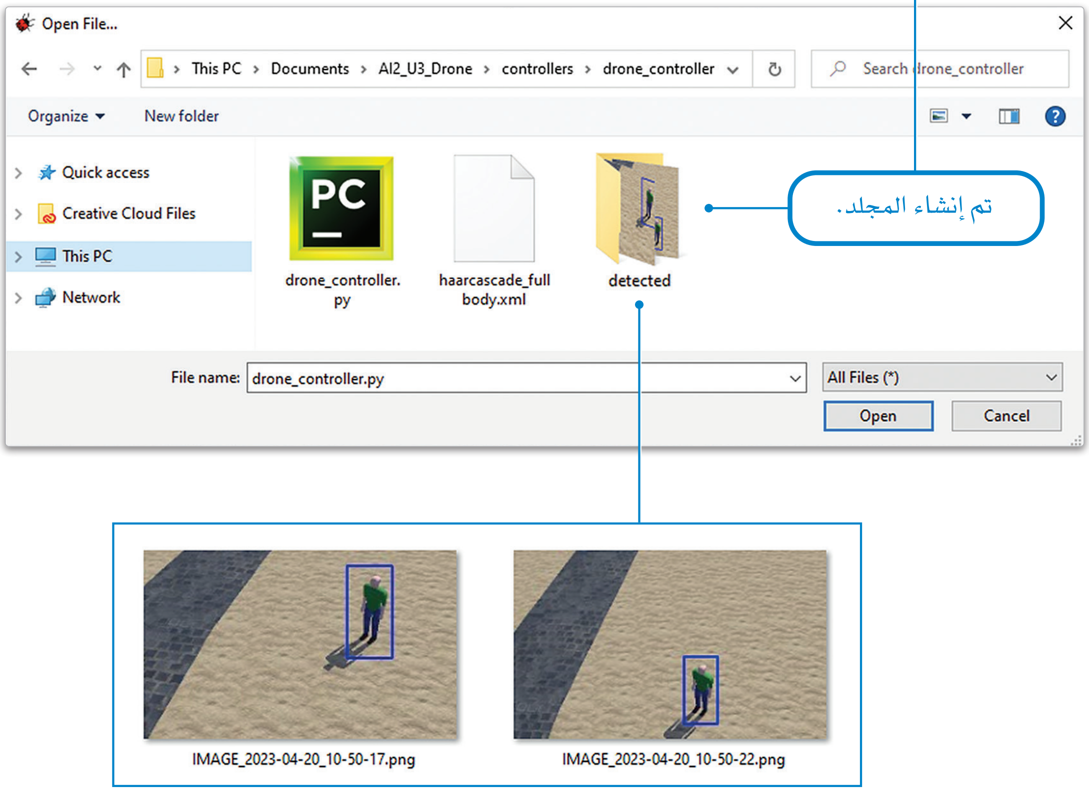
```
def run(self):

    # time intervals used for adjustments in order to reach the target altitude
    t1 = self.getTime()
    # time intervals between each detection for human figures
    t2 = self.getTime()

    roll_disturbance = 0
    pitch_disturbance = 0
    yaw_disturbance = 0

    # specifies the patrol coordinates
    waypoints = [[-30, 20], [-60, 30], [-75, 0], [-40, -10]]
    # target altitude of the drone in meters
    self.target_altitude = 8

    # gets the current working directory
    cwd = os.getcwd()
    # sets the name of the folder where the images
    # with detected humans will be stored
    folder_name = "detected"
    # joins the current working directory and the new folder name
    folder_path = os.path.join(cwd, folder_name)

    if not os.path.exists(folder_path):
    # creates the folder if it doesn't exist already
        os.makedirs(folder_path)
        print(f"Folder \"detected\" created!")
    else:
        print(f"Folder \"detected\" already exists!")

    while self.step(self.time_step) != -1:

        # reads sensors
        roll, pitch, yaw = self.imu.getRollPitchYaw()
        x_pos, y_pos, altitude = self.gps.getValues()
        roll_acceleration, pitch_acceleration, _ = self.gyro.getValues()
        self.current_pose = [x_pos, y_pos, altitude, roll, pitch, yaw]

        if altitude > self.target_altitude - 1:
            # as soon as it reaches the target altitude,
            # computes the disturbances to go to the given waypoints
            if self.getTime() - t1 > 0.1:
                yaw_disturbance, pitch_disturbance = self.move_to_target(
                   waypoints)
                t1 = self.getTime()

        # initiates the image processing and detection routine every 5 seconds
        if self.getTime() - t2 > 5.0:

            # retrieves image array from camera
            cameraImg = self.camera.getImageArray()

            # checks if image is successfully retrieved
            if cameraImg:
```

```
        # reshapes image array to (channels, height, width) format
        cameraImg = np.transpose(cameraImg, (2, 0, 1))
        cameraImg = np.reshape(cameraImg, (3, 240, 400))

        # creates RGB image from merged channels
        img = Image.new('RGB', (400, 240))
        img = cv2.merge((cameraImg[2], cameraImg[1], cameraImg[0]))

        # converts image to grayscale
        gray = cv2.cvtColor(np.uint8(img), cv2.COLOR_BGR2GRAY)

        # loads and applies the Haar cascade classifier to detect humans in image
        human_cascade = cv2.CascadeClassifier('haarcascade_fullbody.xml')
        humans = human_cascade.detectMultiScale(gray)

        # loop, through detected human images, annotates them with a bounding box
        # and prints a timestamp and an info message on the console
        for (x, y, w, h) in humans:

            cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)
            current_time = datetime.now()
            print(current_time)
            print("Found a person in coordinates [{:.2f}, {:.2f}]"
                .format(x_pos, y_pos))

            # saves annotated image to file with timestamp
            current_time = current_time.strftime("%Y-%m-%d_%H-%M-%S")
            filename = f"detected/IMAGE_{current_time}.png"
            cv2.imwrite(filename, img)

        t2 = self.getTime()

# calculates the desired input values for roll, pitch, yaw,
# and altitude using various constants and disturbance values
roll_input = self.K_ROLL_P * clamp(roll, -1, 1)
                                + roll_acceleration + roll_disturbance
pitch_input = self.K_PITCH_P * clamp(pitch, -1, 1)
                                + pitch_acceleration + pitch_disturbance
yaw_input = yaw_disturbance
clamped_difference_altitude = clamp(self.target_altitude
                                - altitude + self.K_VERTICAL_OFFSET, -1, 1)
vertical_input = self.K_VERTICAL_P * pow(clamped_difference_altitude, 3.0)

# calculates the motors' input values based on the desired roll, pitch, yaw, and altitude values
front_left_motor_input = self.K_VERTICAL_THRUST
              + vertical_input - yaw_input + pitch_input - roll_input
front_right_motor_input = self.K_VERTICAL_THRUST
              + vertical_input + yaw_input + pitch_input + roll_input
rear_left_motor_input = self.K_VERTICAL_THRUST + vertical_input
              + yaw_input - pitch_input - roll_input
rear_right_motor_input = self.K_VERTICAL_THRUST + vertical_input
              - yaw_input - pitch_input + roll_input

# sets the velocity of each motor based on the motors' input values calculated above
self.front_left_motor.setVelocity(front_left_motor_input)
self.front_right_motor.setVelocity(-front_right_motor_input)
self.rear_left_motor.setVelocity(-rear_left_motor_input)
self.rear_right_motor.setVelocity(rear_right_motor_input)
```

الآن شغِّل المُحاكاة لترى الطائرة المُسيَّرة وهي تُقلع وتُحلِّق حول المنزل. لاحظ مُخرَجات وحدة التحكم الجديدة والصور التي تم إنشاؤها في المجلد.




شكل 6.21:




شكل 6.22: إنشاء المجلد والصور المحفوظة التي تحتوي على الاكتشافات



وزارة التعليم
Ministry of Education
2025 - 1447

# تمرينات

1 عدِّل برنامج المُتحكِّم الخاص بك بحيث لا يتحقق من وجود المجلد بالفعل في المسار. هل يتسبب ذلك في أية تعقيدات في تنفيذ المُحاكاة؟

2 عدِّل برنامج المُتحكِّم بحيث يقوم بالاكتشاف كل 10 ثوانٍ. هل تلاحظ أي فَرق في تكرار ما تطبعه وحدة التحكم وفي الصور المحفوظة؟

وزارة التعليم
Ministry of Education
2025 - 1447

3 ماذا سيحدث لمُخرَجات الصورة إذا قمت بدمج أبعاد الألوان حسب التسلسل المعتاد بدلًا من التسلسل المعكوس؟ دوِّن ملاحظاتك وفقًا لذلك.

4 أجرِ تجارب على المُعامِلين الرابع والخامس في الدالة ()rectangle. دوِّن ملاحظاتك وفقًا لذلك.

5 عدِّل برنامج المُتحكِّم الخاص بك بحيث يطبع قيم الالتفاف والانحدار والانعراج للطائرة المُسيَّرة عند اكتشاف أي شخص.


وزارة التعليم
Ministry of Education
2025 - 1447

# المشروع

في الوقت الحاضر، هناك العديد من مشاريع تكامل الذكاء الاصطناعي كبيرة الحجم التي يتم تطويرها لمختلف الصناعات والقطاعات المختلفة في البلدان، ويُعدُّ القطاع الصحي من أهم القطاعات التي تتبنى تقنيات الذكاء الاصطناعي، وهذا يعني أن تطوير المشاريع في هذا القطاع لا بُدّ أن يأخذ أخلاقيات الذكاء الاصطناعي بعين الاعتبار.

1 أجرِ بحثًا عن أنظمة الرعاية الصحية التي تعمل بالذكاء الاصطناعي وعن آثارها الأخلاقية، وحدِّد المنافع والمخاطر المحتملة لتطبيق نظام تقنية معلومات يعمل بالذكاء الاصطناعي في مؤسسة صحية.

2 حلِّل المخاوف الأخلاقية التي تنشأ عند استخدام الذكاء الاصطناعي في اتخاذ قرارات تؤثر على صحة المريض، وضَعْ مجموعة من المبادئ الأخلاقية لاستخدام الذكاء الاصطناعي في الرعاية الصحية تعطي الأولوية لسلامة المريض وصحته.

3 أنشئ عرضًا تقديميًا يحدِّد المبادئ الأخلاقية المقترحة والأسباب التي تدعو إلى الالتزام بها، واعرض المبادئ على زملائك في الفصل، ثم ناقش معهم مزايا وتحديات المبادئ المقترحة.

وزارة التعليم
Ministry of Education
2025 - 1447

# ماذا تعلّمت

- معرفة لمحة عامة عن أخلاقيات الذكاء الاصطناعي.
- فحص كيف يُمكن للتحيُّز والافتقار إلى الإنصاف أن يُؤديا إلى إساءة استخدام أنظمة الذكاء الاصطناعي.
- تحديد طرائق التخفيف من مشكلة الشفافية لقابلية التفسير في الذكاء الاصطناعي.
- تقييم كيفية توجيه التنظيمات والمعايير الحكومية للاستخدام الأخلاقي والمستدام لأنظمة الذكاء الاصطناعي.
- برمجة الطائرة المُسيَّرة للتنقل في بيئة ما دون تدخل بشري.
- تعديل نظام الطائرة المُسيَّرة لتشمل قدرات المراقبة من خلال تحليل الصور.

## المصطلحات الرئيسة

| | |
|---|---|
| وحدة قياس بالقصور الذاتي | Inertial Measurement Unit - IMU |
| محرِّك | Motor |
| مكتبة أوبن سي في | OpenCV Library |
| الانحدار | Pitch |
| مروحية | Propeller |
| الروبوتية | Robotics |
| الالتفاف | Roll |
| مُحاكِي | Simulator |
| الاستدلال القائم على القِيم | Value-Based Reasoning |
| الانعراج | Yaw |

| | |
|---|---|
| أخلاقيات الذكاء الاصطناعي | AI Ethics |
| مراقبة المنطقة | Area Surveillance |
| التحيُّز | Bias |
| مشكلة الصندوق الأسود | Black-Box Problem |
| إلغاء الانحياز | Debiasing |
| نظام تحديد المواقع العَالَمي | Global Positioning System - GPS |
| الجيروسكوب | Gyroscope |
| اكتشاف البشر | Human Detection |


وزارة التعليم
Ministry of Education
2025 - 1447